Combining Computation with Geometry

Sheue-Ling-Chang Lien

Computer Science Department
California Institute of Technology

5185:TR:85

Combining Computation with Geometry

Thesis by

Sheue-Ling Chang Lien

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California 91125

5185:TR:85

1985

(Submitted November 30, 1984)

Acknowledgement

Thanks to God for bringing a baby boy and two Ph.D.'s to our family in such a short period of time, and for letting me enjoy the fun of punching the keyboard with one hand and rocking the baby to sleep with the other.

Thanks to my mother for helping me through the difficult time of being a new mother and a Ph.D. candidate at the same time. Thanks to my dear husband for not only being a good father and companion, but also a good collegemate. His encouragement and support helped me very much—not only in my own personal life, but also in my study—especially when I became frustrated from looking at a mound of soiled diapers sitting right next to a stack of unfinished papers.

I am very grateful to my adviser, Dr. James Kajiya, for his patience and guidance. He was willing to grant me the freedom of venturing into an area which is largely foreign to my background. Through his direction, I learned the most important thing: confidence in my ability to identify and address a pertinent research topic. I am also very grateful to Dr. Derek Fender and Dr. George Lewicki. I sincerely appreciate their care.

I would like to thank Young-il Choo, Richard Landry, John Tanner and Brian Von Herzen for helping to polish up this document, and also Peggy Li and Calvin Jackson for helping to print out this document. I am especially grateful to John Tanner for his enthusiasm and helpful discussions. Special thanks go to Calvin Jackson, who is such a nice person, and so willing to help other people.

Abstract

This thesis seeks to establish mathematical principles and to provide efficient solutions to various time consuming operations in computer-aided geometric design. It contains a discussion of three major topics: (1) design validation by means of object interference detection, (2) object reconstruction through the union, intersection, and subtraction of two polyhedra, and (3) calculation of basic engineering properties such as volume, center of mass, or moments of inertia.

Two criteria are presented for solving the problems of point-polygon enclosure and point-polyhedron enclosure in object interference detection. An algorithm for efficient point-polyhedron-enclosure detection is presented. Singularities encountered in point-polyhedron-enclosure detection are categorized and simple methods for resolving them are also included.

A new scheme for representing solid objects, called skeletal polyhedron representation, is proposed. Also included are algorithms for performing set operations on polyhedra (or polygons) represented in skeletal polyhedron representation, algorithms for performing edge-edge intersection and face-face intersection in a set operation, and a perturbation method which can be used to resolve singularities for an easy execution of edge-edge intersection and face-face intersection.

A symbolic method for calculating basic engineering properties (such as volume, center of mass, moments of inertia, and similar integral properties of geometrically complex solids) is proposed. The same method is generalized for computing the integral properties of a set combined polyhedron, and for computing the integral properties of an arbitrary polyhedron in m-dimensional (R^m) space.

Table of Contents

Figures and Tables

Chapter 1

Introduction

From the earliest of times, drawings were the primary medium for communicating geometry between humans. With the advent of computerized drafting, cursors and light pens have replaced pencils, drawings are stored in disks and tapes, and sketches are displayed on ephemeral CRT images. These systems have grown to permit improved productivity, job specialization, and efficient management of design specifications through data bases.

However, there are several things which early computerized drawing systems still cannot do. They cannot, for example, automatically calculate the mass properties of defined objects, or produce perspective views that automatically eliminate hidden lines, or perform a union or intersection operation on two defined objects. These deficiencies are surprising because the tasks themselves are mathematically well-defined, and it is reasonable to expect that they could be performed automatically. The solution to this problem resulted in a new generation of computerized drawing systems, the so-called solid-modeling systems.

"Solid modeling" is a subject which involves the representation, specification, design, manipulation, display, and analysis of free-form curves and surfaces. It encompasses an emerging body of theory, techniques, and systems focused on "informationally complete" representations of solids, that permit any well-defined geometrical property of any represented object to be calculated automatically. This field draws on techniques and principles from several lines of research, including numerical analysis, approximation theory, computer graphics, interactive computer systems, and mechanical and geometric design. We expect this new generation to remove the major roadblock to flexible automation and advance the "A" in CAD/CAM from "aided" to "automated."

1.1 Computer-aided geometric design

It is naturally appealing to develop a system containing full descriptions of the geometric and physical properties of objects and their relationships, on which a mechanical engineer can generate an initial design interactively, perform engineering calculations, and describe the properties and shape of each part. The designer could interact with a 2D display screen using a keyboard, a cursor, and perhaps a light pen to indicate position and parameter values. The screen might contain multiple views with perspective projections. The rendering could follow a selected drawing standard that eliminates hidden lines, and the designer could express a design directly in 3D rather than as multiple 2D projections.

Further, a mechanical engineer would be able to specify such properties as the material type, the heat treatment parameters, and the surface finish of a part. The shape would constitute a complete description of the geometry of a part, including all relevant information, such as mass, surface area, and center of mass. As the design development extended from a single part to multiple parts assembly mechanism, the relationships among parts could be investigated. Static interferences could be checked, dynamic interferences that would occur as parts moved along trajectories could be detected, and mechanisms could be simulated. As the design moved into the manufacturing stage, the same engineering data base could be used again. If a part were to be machined, the numerical control (NC) programs would be generated automatically; if a part were to be cast or molded, the dies would be designed and the NC programs generated.

At any stage of the initial design and subsequent refinement process, a designer could perform design analysis studies appropriate for the level of detail entered thus far. These studies might involve individual objects or design validation in between objects in an assembly, and the analysis might cover computation of basic engineering properties of parts. The basic engineering properties of a geometric object in which we are interested include surface area, volume, center of gravity, and moments of inertia of the object. Each of these may be readily calculated by appropriate integration of

the object. The results of the analysis could be used to confirm the validity of the design or to provide guidance for corrective actions.

Object interference detection is one of the procedures in design validation which investigates geometric interference relationships between parts. The aim is to determine whether parts fit correctly, whether mechanisms move correctly, or whether tolerance distributions are correct. An interactive geometric graphics system allows the designer to define the position and orientation of each object, and to "fly" objects into their correct relative positions. When positioning must be simulated many times, a more powerful technique uses a symbolic description of placement to express the geometric and spatial relationships between objects. This approach allows a procedural description of an assembly sequence; for example, execution of an assembly procedure causes models of the parts to be fetched from a parts library and a scene to be composed with the objects in their correct relative positions. If the designs of objects are changed so that the positions of features are changed, then re-execution of the procedure allows the scene to be recomposed with the parts automatically in their new relative positions.

When simulating the spatial positions of solid objects, it is possible to detect physically unrealizable situations in which objects occupy the same region of space. Object interference detection involves simulating objects in their desired positions and testing to determine whether they are disjoint, whether one object is contained in another, or whether one is partially inside and partially outside the other. It is also possible to test whether an assembly is physically realizable by performing dynamic interference checking: determining whether there is a collision-free trajectory for a part to reach its final position. If no collision-free path can be found, then the assembly cannot be built. This problem then becomes one of testing for collisions between boundaries as objects move–discovering when the edge of one object pierces a face of the other. It is important to notice that although the objects themselves are polyhedral, they may still approximate objects with curved surfaces. Interference tests between polyhedral objects are only meaningful when the error between the polyhedral approximation to curved surfaces is less than the minimum intended distance between parts.

Other features are also required in order to produce a complete description of objects. For instance, a designer should be able to indicate that an entity is part of a complete object, and in addition, whether the part contributes a positive ("solid") or negative ("hole") volume to the object. Complex shapes can be formed by "gluing" or "molding" together simpler shapes (formally speaking, taking the set union or the set intersection of polyhedra), or, conversely, by "cutting and drilling" sections out of parts (taking the set difference of polyhedra). This method greatly reduces the burden of describing all the points, lines and surfaces which define a complex part's polyhedral representation.

1.2 Solid Modeling

Solid modeling is the interactive computer graphics technology used to describe mathematically accurate three-dimensional objects. It has been touted as the technological solution to automating and integrating design and manufacturing functions. The technology is important because of the new dimension it brings to CAD/CAM. The wide range of activities to which the data base can be applied is what makes solid modeling so highly significant.

Mechanical design and manufacturing environments pose specific demands upon the solid modeling technology. Most of the solid modeling systems in mechanical design/manufacturing environments were installed to test the feasibility of integrating these two functions. Recently, however, these systems have started to be shifted toward the design development and structure analysis of actual parts and assemblies. Objects specified and designed using a solid modeling system can be manufactured from a shared database.

Although solid modeling has been used in conjunction with a number of applications including animation production for film-making, the greatest potential for solid modeling is not these displayed images but the ability to provide a more complete part description for design and manufacturing. The technological constraints for a solid modeler for mechanical engineering differ so much from those of the film industry.

Systems capable of developing solid-looking images could be used for the film industry, but they may not be sufficient for using in a mechanical design/manufacturing environment. The major requirement of a solid modeler for mechanical engineering application is to be able to build and contain unambiguous representations of parts or assemblies.

Constructive solid geometry (CSG) and boundary representation (B-rep) are two methods of storing the requisite geometry in a computer system. With CSG, an object is represented by a tree-like data structure that describes how the object is built up from simpler objects. Boolean operations are used to combine the simpler objects. At the bottom of the tree are primitives such as cubes, spheres and cones which constitute the foundation of this building process. Boundary representation is a list of the surfaces, or boundaries, of an object. The advantage of using boundary representation is that solid images can be generated quickly and the solid content of the object can be computed easily. A problem in using this type of construction is ensuring its validity, both combinatorially and geometrically. The advantage of using CSG is that it guarantees the validity of an object, however, it usually requires an enormous amount of time to generate the display of an object.

Other important attributes required by a solid modeling system include such as a friendly user interface, the ability to interact with the model and make modifications, reliable operations within the system, and the ability to apply structural analysis, etc.

1.3 Thesis summary

Today, the increasing speed, capacity, and connectivity of computers and the related advances in display and data entry devices make possible such important engineering advances as direct computation of the area, volume, and mass properties of three-dimensional objects. However, these breakthroughs are governed by both the complexity of the geometry involved and the complexity of computation desired. Besides the task of rewriting known formulas in a programming language, engineers face other significant obstacles in solid representation, data structures, algorithm design, complexity analysis, etc.

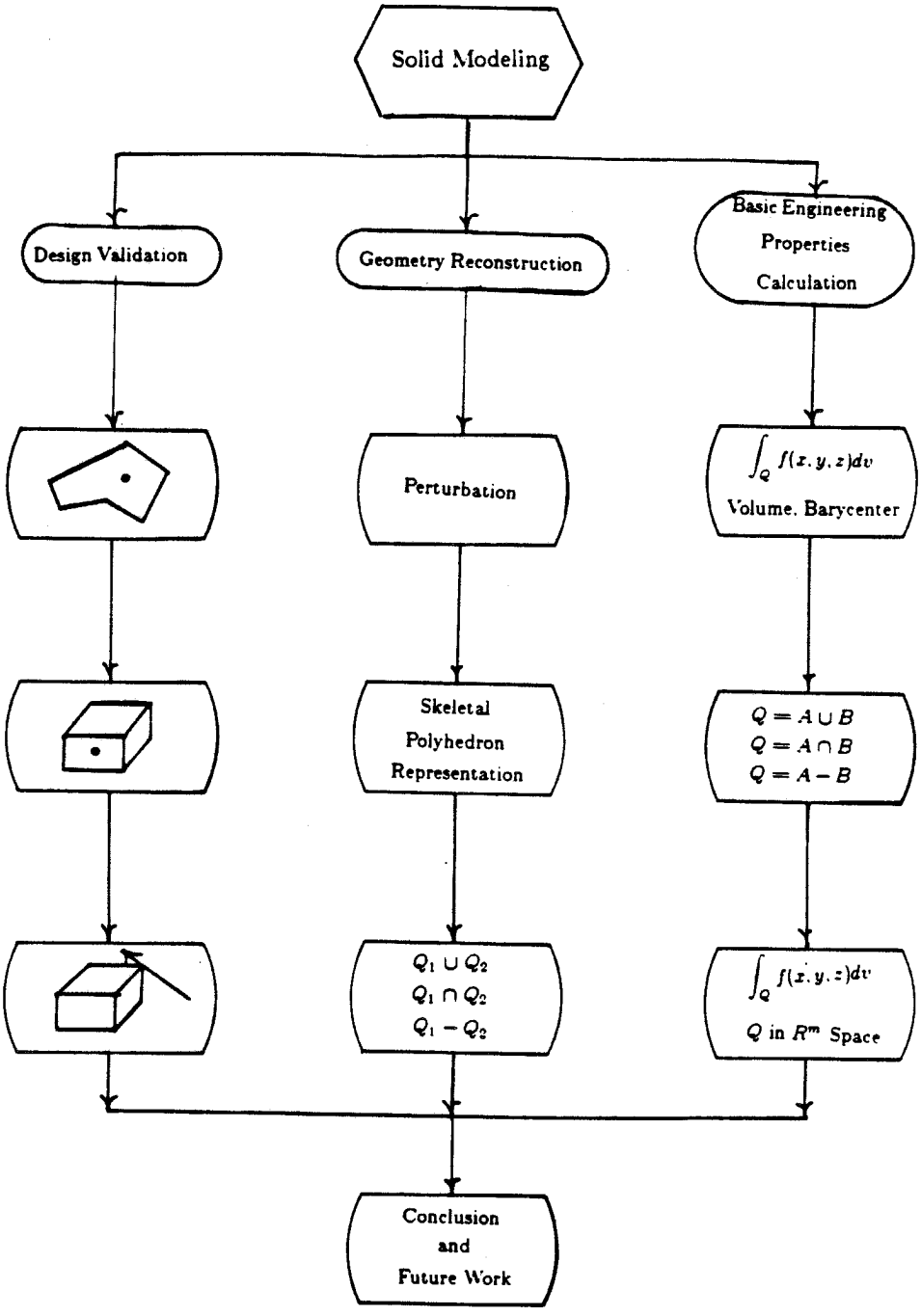

Figure 1

Flowchart of the thesis.

The goal of this research is to contribute to the resolution of some of these issues by exploring powerful means for dealing with geometries. It does this by exploiting basic theorems to strengthen the theoretical foundations of solid modeling; by proposing a new scheme for solid representation; and by developing efficient algorithms for solving some persistent engineering problems and for calculating engineering-related properties. It includes a discussion of three specific topics: (1) design validation by means of object interference detection; (2) object reconstruction through the union, intersection, and subtractions of two polyhedra; and (3) the calculation of basic engineering properties, including volume, center of mass, and moments of inertia of designed objects. Figure 1 describes the organization of this work.

The first part of the thesis (Chapters 2-4) offers a new approach to the common problems of point-polygon enclosure detection and point-polyhedron enclosure detection in design validation. Point-polygon enclosure detection is one of the most time-consuming operations in the elimination of hidden lines on a graphic display; and it is also a major operation of object interference detection in two-dimensional space. Point-polyhedron enclosure detection is a major operation of object interference detection in three-dimensional space. An efficient execution of point-polygon enclosure detection and point-polyhedron enclosure detection can significantly improve both the speed of object interference checking and the performance of the user interface in a solid modeling system.

Chapter 2 proposes a transition criterion for detecting points enclosed in a 2D polygon, which determines the containment relationship by means of as many line equation computations as the number of edges of the polygon. By using the criterion, one does not need to pay special attention to some singular situations which are considered troublesome in using a conventional 2D enclosure detection method. The criterion can also be applied to solving an edge-face-piercing problem, detecting the orientation of a polygon, and detecting whether a point is enclosed by a curve represented with strip-trees.

Chapter 3 presents an efficient algorithm for solving 3D point-polyhedron

enclosure problem. This form of detection, conventionally made by drawing a testing vector from the point to infinity and counting the number of faces penetrated by a testing vector, it is indicated by an odd parity of intersections. The new algorithm improves in efficiency by searching only for the closest face which penetrated by a testing vector; and as a result, it takes on the average, about half the amount of work as the conventional techniques do. In the best case, it reduces the original 3D enclosure problem into only one 2D point-polygon enclosure detection. In the worst case it would take only as many 2D point-polygon enclosure detections as the conventional techniques do.

A singularity in 3D point-polyhedron enclosure detection is a case in which a testing vector coincides with a vertex, an edge, or a face of the polyhedron. The presence of singularities can impair the parity count of intersections. One benefit of the new approach is that it not only handles common cases well, but that it also handles singularities in an efficient procedural fashion. A criterion for solving such cases is proposed in Chapter 4, as a generalization of the transition criterion in Chapter 2. The singularity criterion leads to a simple method for correctly resolving all the singularities possibly encountered, and yet requires only as many plane equation calculations as the number of faces connected to the vertex that induces the singularity. This criterion is applicable to arbitrary nonconvex polyhedra and has no restriction on the number of faces to which a vertex can be connected.

This chapter also categorizes all singularities encountered in 3D point-polyhedron enclosure detection, and includes simple methods for solving them. Singularities are categorized into two basic types, V-singularity and E-singularity, and four complex types, V-V-singularity, V-E-singularity, E-E-singularity, and F-F-singularity. A V-singularity is a case in which a testing vector intersects a vertex of a polyhedron but is not coplanar with any of its converging faces. An E-singularity is a case in which a testing vector intersects an edge of the polyhedron but is not colinear with the edge. A complex singularity is one in which a testing vector is coplanar with a face of the polyhedron, it can always be decomposed into a combination of two basic singularities by bending the testing vector.

The second section (Chapters 5-7) proposes new representational strategies to replace conventional techniques, including algorithms for performing edge-edge intersection and face-face intersection; a perturbation method which can be used to transform two objects into a singularity-free situation for an easy execution of edge-edge intersection and face-face intersection; a new scheme for solid-object representation called skeletal polyhedron representation; and algorithms for performing set operations on solids represented in skeletal polyhedron representation.

The perturbation method proposed in Chapter 5 is a technique used to translate two objects into a singularity-free situation so that a set operation on the two objects can be performed without resorting to the kinds of irregular procedures that singularities conventionally demand. Perturbation is applied in an early stage of edge-edge intersection (in a set operation on two polygons) or face-face intersection (in a set operation on two polyhedra). The advantage of applying this technique at these stages is that singularities can then be resolved independent of the set operation to be performed.

The first step of performing a set operation on two polyhedra is polyhedron-polyhedron intersection. Perturbation is a technique that greatly simplifies this step. Polyhedron-polyhedron intersection compares two polyhedra to find all the intersections between them. It is composed of several independent face-face intersections which compares one face from each polyhedron to determine the intersection between the two faces. Face-face intersection is conventionally solved by decomposing it into multiple edge-face-penetrations and reducing the problem into multiple 2D point-polygon enclosure detections. An efficient technique [9] is used instead which determines the intersection between the two faces by directly merging the ordered cut-edge lists from each face.

Skeletal polyhedron representation discussed in Chapter 6 is a new way of describing solids on the basis of their topological relations. This simple method represents solids by means of the connection and the order between vertices. Boundary faces in this representation are implicit. However, skeletal polyhedron representation

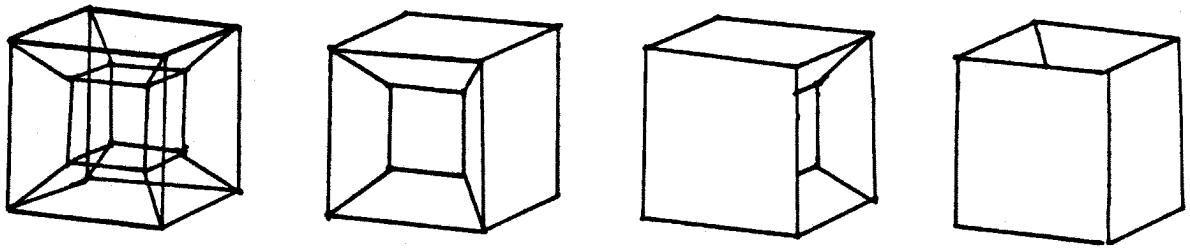

Figure 2
Several similar objects are constructed out of one wire frame representation.

is different from the conventional wire frame representation which describes solids simply by their vertices and edges. As Figure 2 shows, objects described in wire frame representation may be ambiguous.

Chapter 7 discusses skeletal representation for polygons and algorithms for performing set operations on polygons and polyhedra. A planar polygon described in a skeletal representation scheme contains a set of vertex representations; a vertex representation describes the relationship between a vertex and its right- and left-hand neighbors. Although boolean operations on solids can be time-consuming, skeletal polyhedron representation simplifies matters in the case of the set operation in particular, by resolving it into easier local operations.

The last section (Chapters 8-10) offers a particularly efficient algorithm for the calculation of basic engineering properties, such as volume, center of mass, moments of inertia, and similar properties of geometrically complex solids. Volume, center of mass, moments of inertia, etc., are defined by triple integrals over subsets of three-dimensional Euclidean space. Because such quantities figure prominently in static and dynamic simulation equations, where the mass of an object or the effects during rotation must be calculated prior to manufacture, the ability to compute integral properties of geometrically complex solids is an important goal in the field

of CAD/CAM.

A symbolic method for computing integral properties of an arbitrary, possibly nonconvex polyhedron is proposed in Chapter 8. This method works by decomposing a polyhedron systematically into tetrahedra and accumulating integral results from each tetrahedron. An equation is also presented that supplies a direct solution to the integral of a polynomial over a tetrahedron and thereby permits symbolic integration over the polyhedron. This method is analytically exact and applicable to the integral of arbitrary polynomial functions. The time complexity of the method is linearly proportional to the number of vertices of a polyhedron.

A symbolic method for computing the integral properties of a set-combined polyhedron is presented in Chapter 9. A set-combined polyhedron is one which is expressed as the union, intersection, or subtraction of two polyhedra. This method is an extension of the method presented in Chapter 8. It shows how integral properties of a set-combined polyhedron can be calculated without reconstructing the polyhedron by a set operation.

In Chapter 10 the method above is generalized to permit computation of the integral properties of an arbitrary polyhedron in an m-dimensional (R^m) space. This issue features prominently in the fields of linear programming and program analysis mechanization, where the probability that a conditional will yield the value 'true' or 'false' from a path in a program may be determined by the ratio of the volumes of two polyhedra: the polyhedron representing the conjunction of linear inequalities along the path, and the polyhedron representing the boundaries of the variables. Computing the volume of an R^m polyhedron is the key to solving this problem.

Chapter 2

Point-Polygon Enclosure Detection

Testing for point-polygon enclosure plays an important part in many techniques common to computer graphics, including ray tracing [1, 13, 27, 31], hidden-line elimination [28], and constructive solid geometry[4, 22, 26].

In ray tracing, for example, it is necessary to find the intersection of a given ray with a polygon, by finding the point of intersection of the ray with the polygon plane, and then, by testing for enclosure of that point by the polygon.

Similarly, hidden-line elimination procedures must test for point-polygon enclosure when determining whether a vertex of a screen-projected polygon is hidden by a face. This occurs when the vertex and eye lie on opposite sides of the face plane and when the screen-projected point is contained within the polygon.

Finally, constructive solid geometry, it is advantageous to determine point-polygon enclosures quickly when transforming an object representation of a boolean combination of primitives to a polyhedral representation. For example, to intersect two polyhedra, one may compare each face of one polyhedron to every edge of another polyhedron and vice versa [4, 22]. This edge-face-penetration detection involves testing for opposition, intersection locus, and enclosure. The opposition test entails checking whether two ends of an edge are on opposite sides of the plane containing the face; the intersection locus test locates the point where an edge pierces a plane; and the enclosure test detects whether the piercing point is within the face, i.e., whether the edge really pierces the face.

Each of the above procedures requires comparison of a point with a simple polygon, either in two-dimensional or three-dimensional space. In this chapter we present a transition criterion for determining whether a given point lies to the interior or exterior of a polygon. The criterion is described first in two-dimensional space, where a face is simply a two-dimensional simple polygon. Later, we redefine several

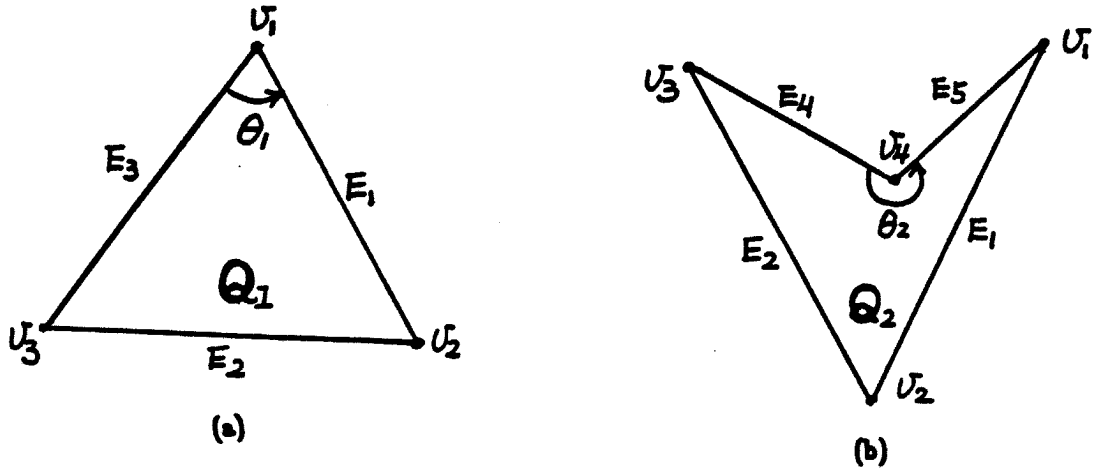

Figure 3
(a) Polygon Q_1 is a convex polygon. Vertex v_1 is a convex vertex. (b) polygon Q_2 is a concave polygon. Vertex v_4 is a concave vertex.

terms used in describing the transition criterion so that we can apply it to edge-face-penetration detection, in which a face is a planar simple polygon in three-dimensional space. In the last section we also describe how this technique can be applied to very complex curves and in determining the orientation of a polygon.

2.1 Notation

Let a simple polygon Q be represented as an ordered list $(v_1, v_2, .., v_n)$ of n vertices. Each pair of consecutive vertices, (v_i, v_{i+1}) in which $i = 1, .., n$, constitutes a directional line segment, edge E_i. No two nonconsecutive edges intersect. Each vertex v_i is represented as a pair $< x_i, y_i >$ where x_i and y_i are its x- and y-coordinates on the plane. The polygon Q divides the plane into interior and exterior regions. For convenience, we assume in this thesis that vertices of a polygon are always ordered in a clockwise direction so that the interior of Q always lies to the right when the edges of Q are traversed.

An edge E_i is a vector from vertex v_i to vertex v_{i+1}. A vertex, say v_i, is

defined as *convex* if the angle between edge E_{i-1} and edge E_i is less than 180 degrees, as with θ_1 in Figure 3a; otherwise it is *concave*, as with θ_2 in Figure 3b. A polygon is defined as convex polygon if it contains no concave vertex. If it does, however, it is called a concave polygon. We can associate with each polygon $Q = (v_1, ...v_n)$, a *vertex pattern* (VP) which describes the convexity and concavity of its vertices:

$$\begin{aligned} VP(Q) &= (vp_1, vp_2, ..., vp_n), \text{ where} \\ vp_i &= 1 \quad \text{if vertex } v_i \text{ is convex} \\ &= -1 \quad \text{if vertex } v_i \text{ is concave} \end{aligned} \tag{2.1}$$

Let $< x_i, y_i >$ and $< x_{i+1}, y_{i+1} >$ be the coordinates of the vertices of an edge $E_i = (v_i, v_{i+1})$. We can describe a line L_i which contains an edge E_i by a line equation $L_i(x, y) = 0$:

$$\begin{aligned} L_i(x, y) &: ax + by + c = 0, \text{ where} \\ a &= y_{i+1} - y_i \\ b &= x_i - x_{i+1} \\ c &= x_{i+1}y_i - x_iy_{i+1} \end{aligned} \tag{2.2}$$

where $< a, b >$ is the normal vector of the edge pointing toward the interior of a polygon. A line divides a plane into two regions: the positive, containing those points satisfying $L_i(x, y) > 0$, and the negative, containing those points satisfying $L_i(x, y) < 0$. We choose the normal vector to point toward the interior of Q so that the interior of Q always lies in the positive region of an edge when edges of Q are traversed. Here regions of an edge are referred to the regions separated by the line containing the edge.

Given a simple polygon $Q = (v_1, v_2, .., v_n)$, and an arbitrary point $P =< x, y >$, we can associate a point P with a *region pattern* (RP), describing in which region of each edge of a polygon Q the point lies. The region pattern bit is $R_i = +$ if P lies in the positive region of edge E_i; $R_i = -$ if P lies in the negative region:

$$\begin{aligned} RP(P, Q) &= (R_1, R_2, ..., R_n), \text{ where} \\ P &=< x, y > \\ R_i &= + \text{ if } L_i(x, y) > 0, \\ R_i &= - \text{ if } L_i(x, y) < 0, \end{aligned} \tag{2.3}$$

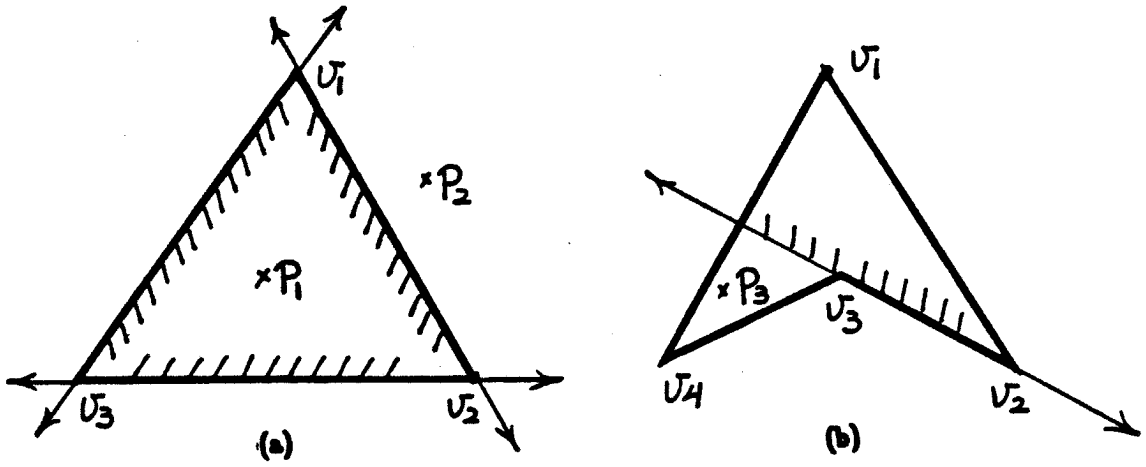

Figure 4
(a) Point P_1 is an interior point and point P_2 is an exterior point of a convex polygon. (b) Point P_3 is an exterior point of a concave polygon.

We know that an interior point of a convex polygon Q must lie in the positive regions of all edges of Q, such as P_1 in Figure 4a. If a point P lies in any one of the negative regions, P must be exterior to Q , such as P_2 in Figure 4a. Therefore, if a polygon is convex, we can easily tell whether a point is inside or outside. However, with a concave polygon, a point may lie in the negative region of an edge, however, still in the interior of polygon, P_3 in Figure 4b. In this case, a nonzero region pattern no longer implies that a point is in the exterior.

The *transition criterion,* permits us to detect in which region of an arbitrary polygon a point lies. We will first use a convex and a concave polygon as examples. Polygon Q_1 in Figure 5 is a triangle composed of three convex vertices v_1, v_2, and v_3, and three edges E_1, E_2, and E_3. The vertex pattern $VP(Q_1)$ equals (111). Three lines L_1, L_2, and L_3 through edges E_1, E_2, and E_3, respectively, split the plane into seven disjoint regions, A, B, C, D, E, F, and G. All points residing in the same region have the same region pattern, and we can associate each region with a unique region pattern. Region A, which is an interior region of Q_1 , has region pattern $RP(A) =$

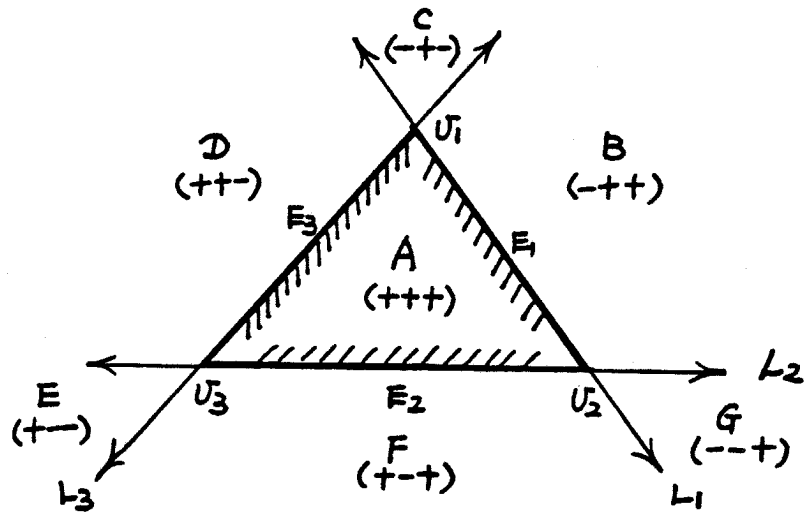

Figure 5

Polygon Q_1 is a triangle with three convex vertices v_1, v_2, and v_3, and three edges E_1, E_2, and E_3. Its vertex pattern is $VP(Q_1) =$ (111); Lines L_1, L_2, and L_3 split the plane into seven disjoint regions, A, B, C, D, E, F, and G. Each region has a unique region pattern.

(+++). Any point residing in region A has the same region pattern (+++). Region patterns of other regions are as follow:

$$RP(A) = (+++), \quad RP(B) = (-++), \quad RP(C) = (-+-), \quad RP(D) = (++-),$$
$$RP(E) = (+--), \quad RP(F) = (+-+), \quad RP(G) = (--+).$$

We define a *transition pattern* (TP) of a point P as the transition state in its region pattern. We say that P has a transition between R_i and R_{i+1} if $R_i \neq R_{i+1}$, i.e., $(R_i, R_{i+1}) = (-+)$ or $(+-)$. A transition pattern of a point P is defined as

$$\begin{aligned} TP(p) &= (T_1, T_2, ..., T_n), \text{ where} \\ T_i &= 0, \quad \text{if} \quad R_{i-1} = R_i, \\ &= 1, \quad \text{if} \quad R_{i-1} \neq R_i \text{ and vertex } v_i \text{ is convex, or} \\ &= -1, \quad \text{if} \quad R_{i-1} \neq R_i \text{ and vertex } v_i \text{ is concave.} \end{aligned} \tag{2.4}$$

If a bit $T_i = \pm 1$, it indicates that P has a transition between edges E_{i-1} and E_i . If $T_i = 0$ we say that P has no transition at vertex v_i , as in Figure 6a; if $T_i = 1$, P has a *convex transition* at v_i , as in Figure 6b; if $T_i = -1$, P has a *concave transition*

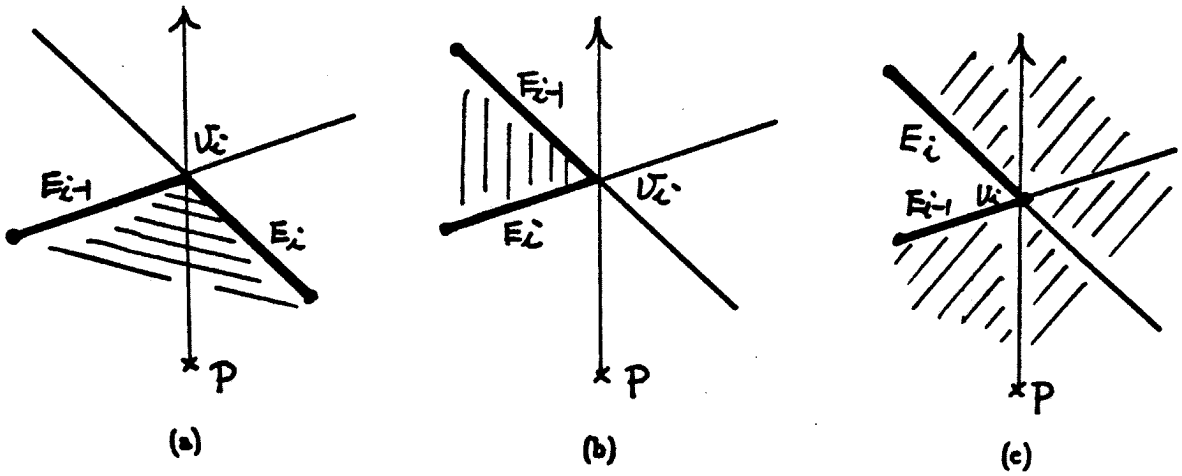

Figure 6
(a) Point P has no transition at vertex v_i, (b) P has a convex transition at v_i, (c) P has a concave transition at v_i.

at v_i , as in Figure 6c. A region is also associated with a unique transition pattern. The transition pattern of a point would also be the transition pattern of the region in which it resides. Region A has $TP(A) = (000)$ since it has no transition. Region B has $TP(B) = (110)$ with two transitions between (R_3, R_1) and (R_1, R_2) and no transition between (R_2, R_3). Following is a list of transition patterns for polygon Q_1 :

$$TP(A) = (000), \quad TP(B) = (110), \quad TP(C) = (011), \quad TP(D) = (101),$$
$$TP(E) = (110), \quad TP(F) = (011), \quad TP(G) = (101).$$

We define a *total-transition* of a point P as

$$\text{Transition}(p) = \sum_{1}^{n} T_i$$

From the above example we can see that region A is an interior region and is the only region having zero transitions. All the others are exterior regions and each has a total transition of two.

Another example is illustrated in Figure 7 where Q_2 is a concave polygon composed of three convex vertices v_1, v_2, v_3, one concave vertex v_4, and four edges

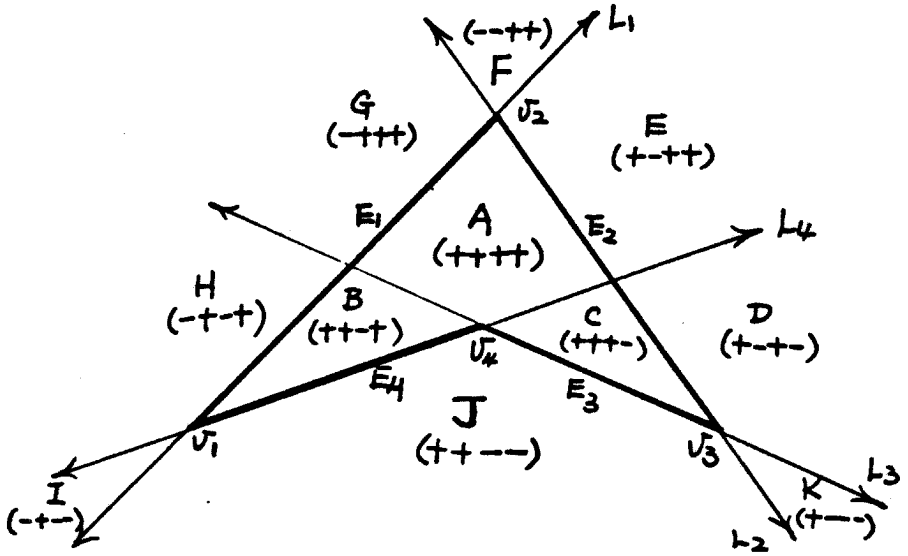

Figure 7

Polygon Q_2 is concave with vertex pattern $VP = (111-1)$.

E_1, E_2, E_3, and E_4. The vertex pattern is $VP(Q_2) = (111{-}1)$. There are eleven disjoint regions, $A, B, ...K$ within Q_2 separated by four lines L_1, L_2, L_3, and L_4 through edges E_1, E_2, E_3, and E_4. The region patterns, transition patterns, and total transitions of Q_2 are listed in Table 1. It shows that regions A, B, and C are interior regions with zero transition; regions, D, E, ... K, are exterior and each has exactly two total transitions.

2.2 A transition criterion for vertex to surface comparison

The transition criterion can be formally described by the following proposition:

I. If a point P resides in the interior of a simple polygon Q, it has an equal number of convex and concave transitions. In other words, it has a total-transition of zero.

II. If a point P resides in the exterior of a simple polygon Q, it has two more convex transitions than concave transitions. In other words, it has a total-

region	RP (region pattern)	TP (transition pattern)	total transitions
A	(++++)	(0000)	0
B	(++−+)	(001-1)	0
C	(+++−)	(100-1)	0
D	(+−+−)	(111-1)	2
E	(+−++)	(0110)	2
F	(−−++)	(1010)	2
G	(−+++)	(1100)	2
H	(−+−+)	(111-1)	2
I	(−+−−)	(0110)	2
J	(++−−)	(1010)	2
K	(+−−−)	(1100)	2

Table 1

The region patterns, transition patterns, and total transitions of Q_2

transition of two.

The remainder of the section presents a proof of the transition criterion. However, the proof is not crucial to understanding the rest of the chapter. The proof of the transition criterion proceeds in the following way:

Step 1. An ***internal triangle*** of a polygon is a triangle formed by three vertices of the polygon and located completely within the polygon. A simple polygon can always be decomposed into a set of mutually disjoint internal triangles.

Step 2. A simple polygon can be rebuilt by recomposing the internal triangles. During each recomposition, two partly built polygons share only one edge.

Step 3. A triangle has already been proved in the second section to satisfy the transition criterion.

Step 4. If a simple polygon $Q = (v_1, .., v_{m+n-2})$ is formed by the union of two semi-disjoint simple polygons $Q_a = (v_a^1, .., v_a^m)$ and $Q_b = (v_b^1, .., v_b^n)$ where Q_a and Q_b share one edge and Q_a and Q_b both satisfy the transition criterion, polygon Q also satisfies the transition criterion.

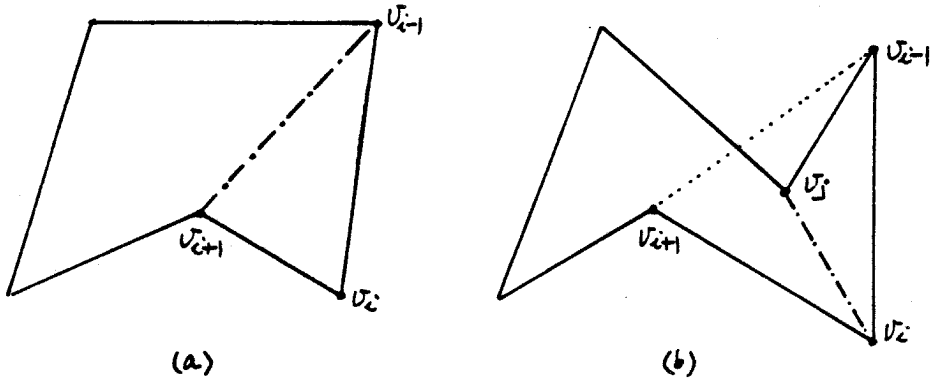

Figure 8
(a) The line segment (v_{i-1}, v_{i+1}) splits a polygon into two smaller non-overlapping polygons. (b) The line segment (v_i, v_j) splits a polygon into two smaller nonoverlapping polygons.

To prove that a simple polygon can always be decomposed into a set of internal triangles, we show that within an arbitrary simple polygon $Q = (v_1, .., v_n)$ with n>3, we can find at least a pair of nonconsecutive vertices v_i and v_j, so that the line segment S_{ij} connecting the two vertices is located completely in the polygon and splits polygon Q into two smaller nonoverlapping simple polygons. Since there is at least one convex vertex v_i within an arbitrary simple polygon Q, a triangle $T = (v_{i-1}, v_i, v_{i+1})$ can be formed with its two neighboring vertices v_{i-1} and v_{i+1}. If none of the other vertices of Q is inside triangle T, triangle T is completely in the interior of polygon Q. In this case the line segment $S_{(i-1)(i+1)}$, connecting vertices v_{i-1} with v_{i+1}, is the segment which splits polygon Q into two smaller simple polygons, Figure 8a. If there are vertices located inside the triangle T, the line segment S_{ij} connecting v_i to the closest vertex v_j is the one which splits the polygon Q into two smaller simple polygons, as in Figure 8b. Polygon Q can be split along line segment S_{ij} into two smaller simple polygons. By induction, the polygon can be decomposed into a set of internal triangles where no two triangles overlap.

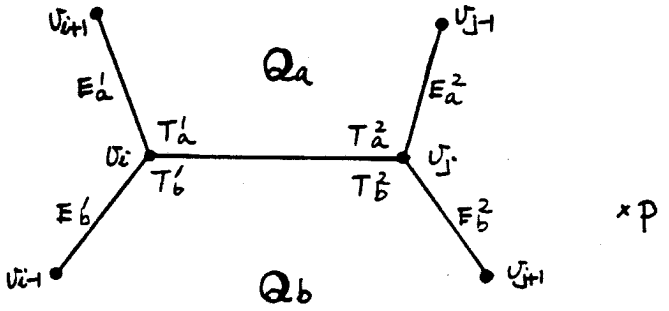

Figure 9

Polygon Q is the union of Q_a and Q_b. T_a^1 and T_a^2 are the transition pattern bits of P in polygon Q_a in between edges (E_a^1, E) and (E, E_a^2), respectively. T_b^1 and T_b^2 are the transition pattern bits of P in Q^b in between edges (E_b^1, E) and (E, E_b^2), respectively. T_1 and T_2 are the transition pattern bits of P with respect to the new polygon in between edges (E_a^1, E_b^1) and (E_a^2, E_b^2), respectively.

Let P be a testing point and $Q_a = (v_a^1, .., v_a^m)$ and $Q_b = (v_b^1, .., v_b^n)$ be two simple polygons satisfying the transition criterion. A new simple polygon Q is formed by the union of Q_a and Q_b under the following two conditions :

I : Q_a and Q_b share exactly one edge;

II : the interiors of Q_a and Q_b are disjoint.

Let edge $E = (v_i, v_j)$ represent the edge shared by Q_a and Q_b. The new polygon $Q = (v_1, .., v_{m+n-2})$, as shown in Figure 9, contains a total of $m + n - 2$ vertices, where each vertex belongs to either Q_a or Q_b, with the exception of vertices v_i and v_j which belong to both. Edges $E_a^1 = (v_i, v_{i+1})$ and $E_a^2 = (v_{j-1}, v_j)$ are the neighboring edges of E within the polygon Q_a . T_a^1 and T_a^2 are the transition pattern bits of P with respect to edge pairs (E_a^1 , E) and (E , E_a^2), respectively. Edges $E_b^1 = (v_{i-1}, v_i)$ and $E_b^2 = (v_j, v_{j+1})$ are the neighboring edges of E within polygon Q_b, and T_b^1 and T_b^2 are

the transition pattern bits of P with respect to $(E_b^1\ ,\ E\ )$ and $(E\ ,\ E_b^2\ )$ respectively. As the new polygon Q is concerned, T_1 and T_2 are the transition pattern bits of P with respect to $(E_a^1\ ,\ E_b^1\ )$ and $(E_a^2\ ,\ E_b^2\ )$, respectively.

The transition criterion can be easily proved by following three fundamental theorems.

1. (Combination Requirement):

For two polygons Q_a and Q_b to be combined under conditions I and II, the transition pattern bits with respect to Q_a and Q_b on both ends of the shared edge must satisfy the constraints:

$$T_a^1 + T_b^1 \geq 0 \text{ and } T_a^2 + T_b^2 \geq 0$$

2. (Local Transition Reservation):

If polygon Q is the union of Q_a and Q_b under conditions I and II, the transition pattern bits with respect to Q , Q_a, and Q_b on both ends of the shared edge must satisfy the equations:

$$\begin{cases} T_1 &= T_a^1 + T_b^1 - 1 \\ T_2 &= T_a^2 + T_b^2 - 1 \end{cases} \tag{2.5}$$

3. (Combinatorial Transition Reservation): If polygon Q is the union of Q_a and Q_b under conditions I and II, the total transitions with respect to Q , Q_a, and Q_b must satisfy the equation:

$$TT = TTa + TTb - 2, \qquad \text{where}$$
$$TTa = \sum_1^m T_a^i, \quad TTb = \sum_1^n T_b^i, \quad \text{and } TT = \sum_1^{m+n-2} T_i \tag{2.6}$$

In the above equation TT, TTa, and TTb denote the total transitions of point P with respect to polygons Q , Q_a , and Q_b respectively.

For two polygons to be combined under the two conditions listed above, the *combination requirement* placed on the transition bits is that the summation of the transition bits of the shared vertex must be greater than zero. If the transition bits on the shared vertex are as listed in cases (7), (8), or (9) in Table 2, then the two polygons cannot be combined under the two conditions.

	T_a^1	T_b^1	T_1
1	1	1	1
2	1	0	0
3	0	1	0
4	0	0	−1
5	1	−1	−1
6	−1	1	−1
7	0	−1	nonexist
8	−1	0	nonexist
9	−1	−1	nonexist

T_a^1 is the transition pattern bit of P with respect to polygon Q_a in between edges E_a^1 and E.
T_a^2 is the transition pattern bits of P with respect to Q_a in between edges E and E_a^2.
T_1 is the transition pattern bits of P with respect to polygon Q in between edges E_a^1 and E_b^1.

Table 2

We prove the *local transition reservation* simply by listing the transition pattern bits T_a^1, T_b^1, and T_1 in Table 2 to show that $T_1 = T_a^1 + T_b^1 - 1$ is always true. The transition pattern bit T_2 combined from T_a^2 and T_b^2 also follows the same rule.

Next we prove the relation of *combinatorial transition reservation* by showing that (1) for a vertex v_k of Q which originally belongs to Q_a but not to Q_b , the transition pattern bits are the same for Q as for Q_a, i.e., $T_k = T_a^k$; and this statement also holds true when v_k originally belongs to Q_b but not to Q_a ; (2) for the two vertices v_i and v_j shared by Q_a and Q_b , the transition pattern bit T_1 and T_2 must satisfy that $T_1 = T_a^1 + T_b^1 - 1$ and $T_2 = T_a^2 + T_b^2 - 1$; (3) the total transition of Q , therefore, satisfies $TTa + TTb - 2$.

We then prove the transition criterion by the *combinatorial transition reservation* theorem, showing that, given two polygons satisfying the criterion, a new polygon combined from them still satisfies the criterion. Assuming that P is in one of the

polygons, say Q_a, then the total transition of P on Q_a equals zero and that on Q_b equals two. Through the combinatorial transition reservation theorem, the total transitions of P on Q is zero, i.e., $TT = TTa + TTb - 2 = 0$. Since P is in the interior of Q_a, then P is in the interior of the combined polygon Q, and the above result satisfies the transition criterion. If P is in neither Q_a nor Q_b, then P has two transitions on both Q_a and Q_b , i.e., $T_a^1 = T_b^1 = 2$, and P is in the exterior of the combined polygon Q. Therefore, it is proved again that the total transitions of P on Q is two (i.e., $TT = TTa + TTb - 2 = 2$). this also satisfies the transition criterion.

2.3 Inverse polygon

An inverse polygon is one in which the finite region confined by the perimeter of the polygon is the "exterior" and the remaining infinite region is the "interior". The transition criterion for an inverse polygon can be described as follows:

I. If a point P is enclosed in an inverse simple polygon Q, that is if P resides in the exterior of the polygon, the point has an equal number of convex and concave transitions. In other words, it has a total-transition of zero.

II. If a point P is not enclosed in an inverse simple polygon Q , that is P resides in the interior of the polygon, it has two more *concave* than *convex* transitions. In other words, it has a total-transition of -2.

Polygon Q in Figure 10 is an inverse polygon with three concave vertices. Its vertex pattern is $VP(Q) = (-1 - 1 - 1)$. The equations of lines L_1', L_2', and L_3' containing the edges E_1', E_2', and E_3' respectively, are defined opposite manner from the line equations in Figure 5, i.e., $L_1'(x, y) = -L_1(x, y)$, $L_2'(x, y) = -L_2(x, y)$, and $L_3'(x, y) = -L_3(x, y)$.

The region pattern and the transition pattern of the seven regions divided by L_1', L_2', and L_3' can also be defined as previously described. It is easy to see that only the three exterior regions have zero total transition; each of the other regions has a total transition of minus two.

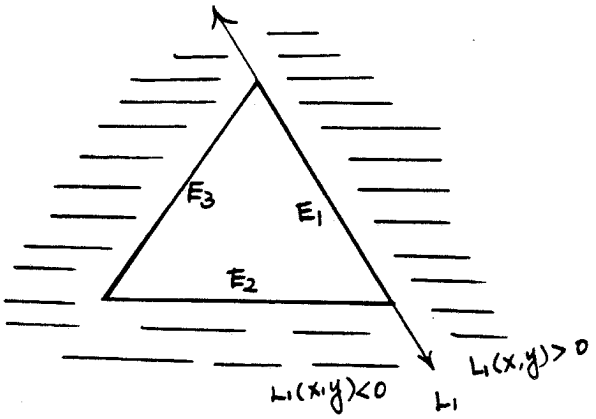

Figure 10

An inverse triangle.

2.4 Degenerate cases

We have excluded the degenerate cases during the above discussion by assuming that every region pattern bit can always be defined. However, in the case where a testing point is standing right on an edge or on the extension line of an edge, the line equation value is zero and that particular region pattern bit cannot be defined. Such situations are referred as degenerate cases. The way to solve a degenerate case is to first discover whether the testing point is standing right on the edge or simply on the extension line of an edge. In the former case the testing point is located on the polygon, and such a situation can be easily detected. In the latter case, where a testing point is located on the extension line of an edge but not directly on the edge, we can always substitute the region pattern bit with a positive sign so that the transition criterion can still be applied.

Figure 11a shows a deliberately produced case where the testing point P is located on the extension lines of four edges. The line equation values of P with respect to each of the above four edges, E_2, E_4, E_6 and E_8, each equals zeros. However, by substituting the region pattern bits of E_2, E_4, E_6 and E_8 with positive signs, we transform the problem virtually into the case as shown in Figure 11b. In this situation

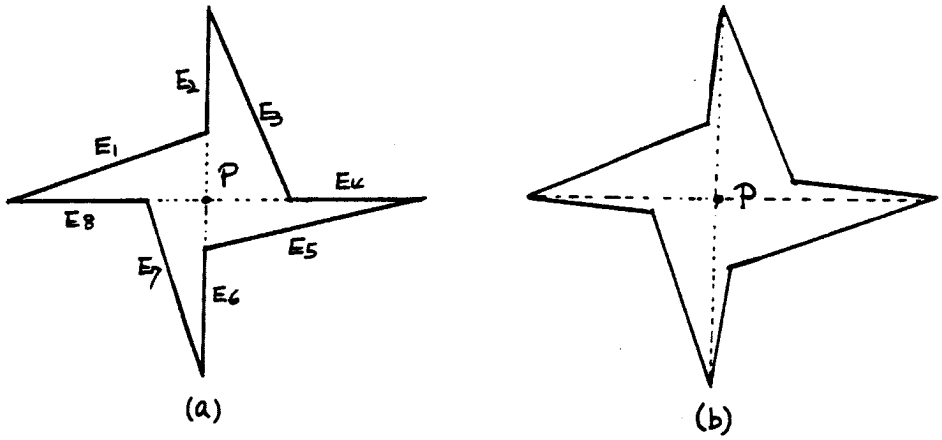

Figure 11
(a) Point P is located on the extension lines of edges E_2, E_4, E_6 and E_8. (b) The polygon is virtually perturbed so that P does not lie on the extension lines of the edges.

there are no more degenerate cases, and the original point-polygon enclosure problem can still be correctly resolved with the result from the virtual case. Therefore, the transition criterion is not crippled by the degenerate cases.

2.5 Applications

2.5.1 Point-polygon enclosure detection

We show a more complicated example in Figure 12, where polygon Q has seven convex vertices, four concave vertices, and eleven edges. Polygon Q has vertex pattern $VP(Q) = (1, -1, 1, 1, 1, -1, -1, 1, 1, -1, 1)$ in which v_2, v_6, v_7, and v_{10} are concave vertices. Point P_1 has a region pattern $RP(P_1) = (-++---+++-+)$, and transition pattern $TP(P_1)$=$(1, -1, 0, 1, 0, 0, -1, 0, 0, -1, 1)$. Point P_1 has three convex transitions at v_1, v_4, and v_{11}, and three concave transitions at v_2, v_7, and v_{10}. We can see that P_1 has a total-transition of zero, implying that P_1 is in the interior of Q . Point P_2 has region pattern $RP(P_2) = (-+-+-+-++-+)$, and transition pattern

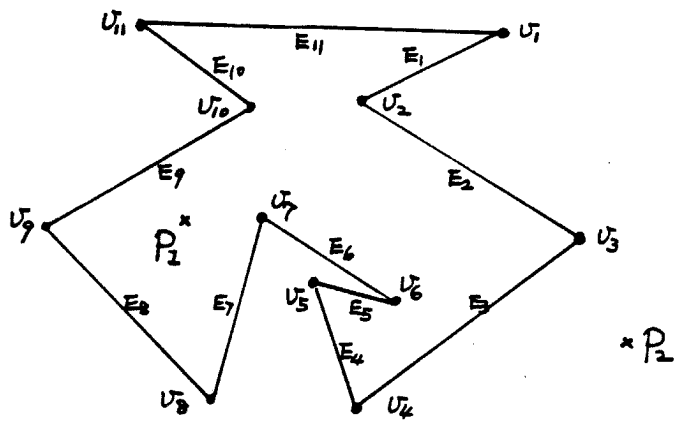

Figure 12
Point P_1 has region pattern $RP(P_1) = (-++---+++-+)$. Point P_2 has region pattern $RP(P_2) = (-+-+-+-++-+)$.

$TP(P_2) = (1, -1, 1, 1, 1, -1, -1, 1, 0, -1, 1)$. P_2 has a total-transition of two, including six convex transitions at $v_1, v_3, v_4, v_5, v_8, v_{11}$, and four concave transitions at v_2, v_6, v_7, and v_{10}. According to the transition criterion, P_2 is exterior to Q .

2.5.2 Three-dimensional edge-face-penetration detection

There are three steps in edge-face-intersection detection: *opposition, intersection, and enclosure.* Given a face F and an edge E, the opposition test determines whether both ends p_1 and p_2 of an edge E are in opposite half-spaces with respect to the plane G containing face F. An edge E is represented by its two terminals $E = (p_1, p_2)$ with coordinates $p_1 =< x_1, y_1, z_1 >$, and $p_2 =< x_2, y_2, z_2 >$. A plane G containing F can be represented as a four-element vector $G = (a, b, c, d)$ in the homogeneous space. If the following plane equation values $G(P_1)$ and $P_2.G$ have opposite signs, i.e., $G(P_1)G(P_2) < 0$, we know that points P_1 and P_2 are in opposite half-spaces of plane G. we know that edge E pierces through plane G.

$$G(P_1) = P_1.G = a.x_1 + b.y_1 + c.z_1 + d,$$
$$G(P_2) = P_2.G = a.x_2 + b.y_2 + c.z_2 + d.$$

The intersection test locates the point where an edge pierces a plane G. First we represent a point P on a line in between points P_1 and P_2 by a parametric equation:

$$\begin{aligned} P &= P_1 + \alpha(P_2 - P_1), \\ &=< x_1 + \alpha(x_2 - x_1), \quad y_1 + \alpha(y_2 - y_1), \quad z_1 + \alpha(z_2 - z_1) >, \\ &\text{where } \alpha \text{ is a parameter, and } 0 <= \alpha <= 1 \end{aligned} \tag{2.7}$$

A piercing point can be found by means of interpolating between P_1 and P_2. Since the piercing point P is located in plane G, the plane equation value equals zero:

$$a[x_1 + \alpha(x_2 - x_1)] + b[y_1 + \alpha(y_2 - y_1)] + c[z_1 + \alpha(z_2 - z_1)] + d = 0 \tag{2.8}$$

We can solve equation (2.8) for α :

$$\alpha = G(p_1)/(G(p_1) - G(P_2)) \tag{2.9}$$

The enclosure test determines whether an edge really pierces the interior of a face; in other words, tells whether the piercing point is within the face. However, the transition criterion discussed so far applies only to simple polygons in two-dimensional space. In face-piercing detection, a face is a planar simple polygon in three-dimensional space and so is a piercing point. The transition-criterion is still applicable to the face-piercing detection, after redefining simple polygons in a three-dimensional space.

A 3D simple polygon is a planar polygon located in three-dimensional space. A 3D polygon Q can be represented as an ordered list $(v_1, v_2, .., v_n)$ where $v_1, v_2, .., v_n$ are a sequence of coplanar vertices in 3D space. Each vertex v_i for $i = 1, .., n$, is represented as $< x_i, y_i, z_i >$ where x_i, y_i and z_i are x-, y-, and z-coordinates in the space. A 3D polygon Q also divides the plane where it is located into interior and exterior regions.

A line L_i through an edge $E_i = (v_i, v_{i+1})$ also divides the plane G into positive and negative regions. We assume that vertices of a face are always listed in a clockwise order when the normal vector of the face points outward from the polyhedron so that the interior of the face always lies to the right half-plane of an edge. Nevertheless, we are left with the problem of deciding in which region with respect to a line L_i ,

positive or negative, an arbitrary point P in plane G lies. This can be solved by taking an outer product $OUP(P, v_i, v_{i+1})$ of vector (P, v_i) with vector (P, v_{i+1}), which is a rotation axis from vector (P, v_i) to (P, v_{i+1}) and is normal to plane G. If point P is in the positive half-plane with respect to line L_i , vector OUP is in the same direction as the normal vector of the face. Otherwise, as the following set of equations show, OUP is in the opposite direction.

$$\begin{aligned} P &= < x, y, z >, \\ v_i &= < x_i, y_i, z_i >, \\ v_{i+1} &= < x_j, y_j, z_j >, \\ N &= (n_x, n_y, n_z), \text{ normal vector of plane } G \end{aligned}$$

$$\begin{aligned} (P, v_i) &= (x_i - x, y_i - y, z_i - z) \text{ vector from } P \text{ to } v_i \text{ ,} \\ &= (v_{xi}, v_{yi}, v_{zi}), \\ (P, v_{i+1}) &= (x_j - x, y_j - y, z_j - z) \text{ vector from } P \text{ to } v_{i+1} \text{ ,} \\ &= (v_{xj}, v_{yj}, v_{zj}), \end{aligned} \tag{2.10}$$

$$\begin{aligned} OUP(P, v_i, v_{i+1}) &= \text{ Outer product of vectors } (P, v_i) \text{ with } (P, v_{i+1}) \\ &= (v_{yi}.v_{zj} - v_{yj}.v_{zi}, v_{zi}.v_{xj} - v_{zj}.v_{xi}, v_{xi}.v_{yj} - v_{xj}.v_{yi}), \\ &= (ou_x, ou_y, ou_z). \\ INP(P, v_i, v_{i+1}) &= \text{ Inner product of } OUP(P, v_i, v_{i+1}) \text{ with normal vector of plane } G \\ &= (n_x.ou_x, n_y.ou_y, n_z.ou_z). \end{aligned}$$

In the former case where point P is in the positive region of edge E_i with respect to plane G, the inner product of OUP with the plane normal vector is positive; in the latter case where P is in the negative region, it is negative. The vertices of a polygon are always in the clockwise order when the normal vector of plane G points outward, so that the interior of Q always lies in the positive region of an edge when edges of Q are traversed. Based on the definition above, the region pattern of a point $P = < x, y, z >$ corresponding to a 3D simple polygon Q can be defined in the same way :

$$\begin{aligned} RP(p) &= (R_1, R_2, ..., R_n), \text{ where} \\ R_i &= + \text{ if p lies in the positive region of edge } E_i \text{ with respect to plane } G, \\ &= - \text{ if p lies in the negative region of edge } E_i \text{ with respect to plane } G, \\ \text{for } i &= 1, .., n \end{aligned}$$

(2.11)

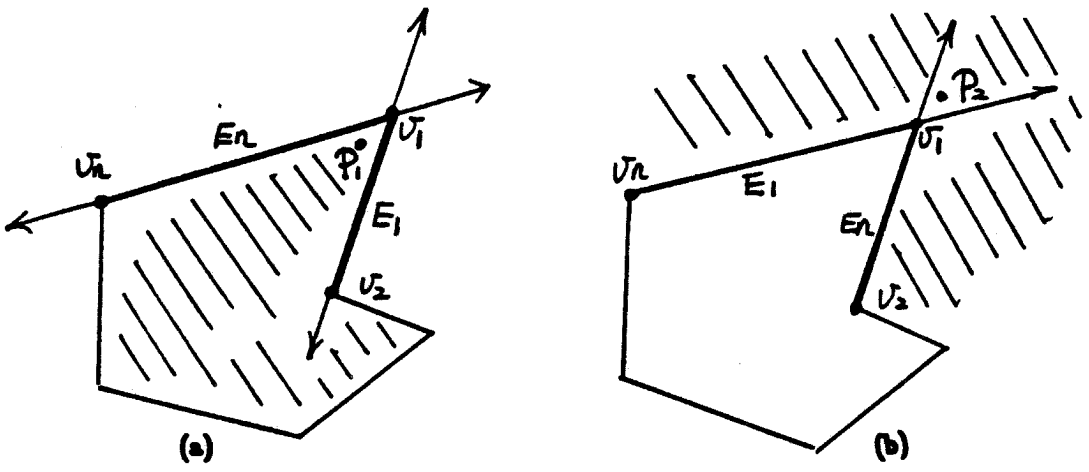

Figure 13

(a) A ordinarily oriented polygon. (b) An inversely oriented polygon.

We can also define the vertex pattern, the transition pattern, and the interior-exterior pattern with respect to a 3D simple polygon as before. By redefining all of these patterns, we can apply the transition criterion described in section 2 to vertex-face comparison in three-dimensional space and as well as to face-piercing detection.

2.5.3 Polygon orientation detection

The interior of an ordinary polygon is assigned to the finite region confined by its perimeter. With an inverse polygon, however, the finite confined region is as the exterior. Generally, these two polygons are distinguished by describing their vertices in either a clockwise or a counterclockwise order. However, partial edge information is not enough to determine the orientation of a polygon. The transition criterion permits us to distinguish the orientation of a polygon in an easy way.

For instance, assuming that the orientation of the polygon $Q = (v_1, v2, .., v_n)$ in Figure 13 is unknown, a virtual testing point P can be set up barely next to the first vertex v_1; that is, the coordinates of P are approximately the coordinates of v_1. The region pattern of P with respect to Q can be calculated by substituting the coordinates $v_1 =< x_1, y_1 >$ into the line equations of each edge of Q . However, the line equation

values of P with respect to E_1 and E_n are zero, i.e., $L_1(x_1, y_1) = 0$ and $L_n(x_1, y_1) = 0$, since v_1 is the intersection of edges E_1 and E_n. We assume that the testing point P is slightly displaced into the (++) corner of v_1, that is the line equation values of P with respect to E_1 and E_n are substituted by (++). The region pattern of P with respect to Q can then be fully determined. If the polygon is ordinarily oriented, the testing point is enclosed by the polygon, as in Figure 13a. According to the transition criterion, P has a total-transition of zero. In case the polygon is inversely oriented, the testing point is excluded by the polygon, Figure 13b. According to the transition criterion, the testing point has a total-transition of minus-two.

2.5.4 Very complex curves

Lastly, the transition criterion is adapted to curves represented in strip trees. A strip tree is a hierarchical representation for a planar curve, consisting of a binary tree with a special datum called a strip at each node, [2]. Lower levels in the tree correspond to finer resolution representations of the curve. Strip tree structure is a direct consequence of using a special method to digitize lines while retaining all the intermediate steps; this permits curves to be efficiently encoded and displayed at various resolutions. A strip tree S is defined to be a six-tuple (v_b, v_e, w_r, w_l) where $v_b =< x_b, y_b >$ denotes the beginning of the strip, $v_e =< x_e, y_e >$ denotes the end, and w_r and w_l, respectively, denote the right and left distances of the strip borders from the directed line segment (v_b, v_e). A curve is approximated by a polygonal line with an ordered list of discrete vertices $(v_1, ..., v_n)$.

We can represent a simple connected area by means of a strip tree that describes its border, a planar closed curve enclosing the area and having the same starting and end point. With the application of the transition criterion, we can determine in a straightforward manner if a point lies inside the area. By taking further advantage of the hierarchical structure of strip tree representation, we can compute the enclosure test very efficiently.

Figure 14a shows a simple connected area represented by a strip tree. The first level in the tree consists of four strips, S_1, S_2, S_3, and S_4, each representing a

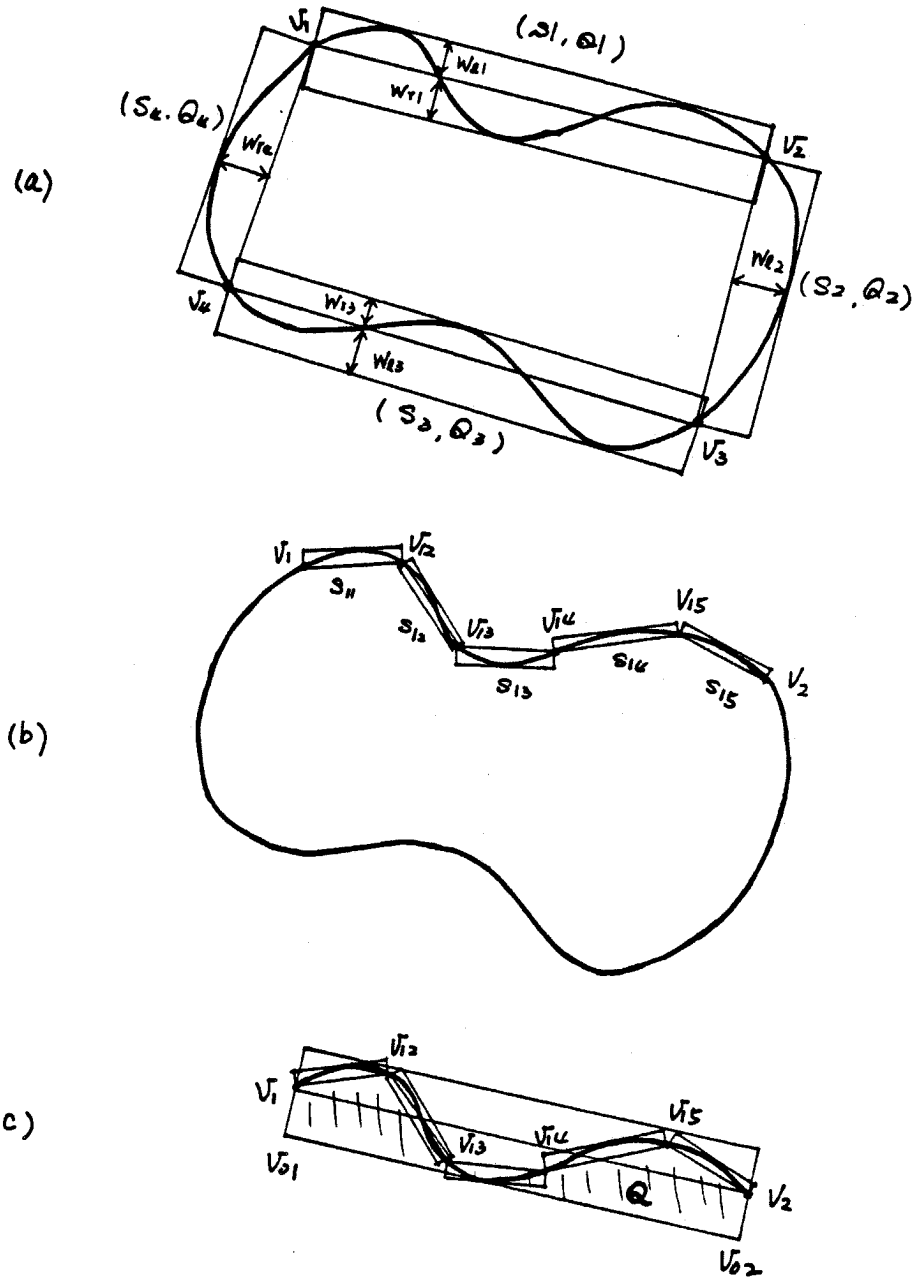

Figure 14

(a) A simple connected area represented by a strip tree. (b) The strip S_1 is decomposed into several lower level strips. (c) If a point is located in strip S_1, it must be in either one of the five rectangles, Q_{11}, ..., Q_{15}, or in the central polygon $Q = (v_{01}, v_1, v_{12}, v_{13}, v_{14}, v_{15}, v_2, v_{02})$.

portion of the border such that:

$$S_1 = (v_1, v_2, w_{r1}, w_{l1}),$$
$$S_2 = (v_2, v_3, w_{r2}, w_{l2}),$$
$$S_3 = (v_3, v_4, w_{r3}, w_{l3}),$$
$$S_4 = (v_4, v_1, w_{r4}, w_{l4}).$$

Each strip can be decomposed into several lower-level strips with finer resolution, as Figure 14b shows. The strips here are *regular strips* as defined in [2], with endpoints touching the ends of the curve. Four rectangles, q_1, q_2, q_3, and q_4 are also defined, containing strips s_1, s_2, s_3 and s_4, respectively. Each rectangle q_i is oriented with vector (v_i, v_{i+1}), and has endpoints v_i and v_{i+1}, length$= |(v_i, v_{i+1})|$, and width$= (w_{ri} + w_{li})$.

To determine whether a point P is inside the area, we can apply the transition criterion to the execution of the following procedures

Step 1: Detect whether P is in either one of the four rectangles.

Step 2: If not, detect whether P is in the central polygon (v_1, v_2, v_3, v_4).

Step 3: (a1). If point P is not in one of the rectangles, or in the central polygon (v_1, v_2, v_3, v_4), point P is *outside* the area and the computation is terminated.

Step 4: (a2). If point P is not in any of the rectangles, but in the central polygon (v_1, v_2, v_3, v_4), point P is *inside* the area and the computation is terminated.

Step 5: (a3). If point P is in one of the rectangles, e.g., rectangle q_1, go down one level on the strip tree and repeat the above procedure.

The curve contained in strip s_1 is decomposed into five lower level strips, $s_{11}, s_{12}, s_{13}, s_{14}$, and s_{15}, each representing a smaller segment of the border:

$$s_{11} = (v_1, v_{12}, w_{r11}, w_{l11}),$$
$$s_{12} = (v_{12}, v_{13}, w_{r12}, w_{l12}),$$
$$s_{13} = (v_{13}, v_{14}, w_{r13}, w_{l13}),$$
$$s_{14} = (v_{14}, v_{15}, w_{r14}, w_{l14}),$$
$$s_{15} = (v_{15}, v_2, w_{r15}, w_{l15}).$$

Again five rectangles, $q_{11}, q_{12}, q_{13}, q_{14}$, and q_{15} are defined to enclose strips $s_{11}, s_{12}, s_{13}, s_{14}$, and s_{15}. To determine whether the point P, as known in rectangle q_1, is inside the area, we then execute the same procedures as above:

Step 1: Detect whether P is in either one of the five rectangles, q_{11}, q_{12}, q_{13}, q_{14}, or q_{15}. If not, then:

Step 2: Detect whether P is in the central polygon $(v_{o1}, v_1, v_{12}, v_{13}, v_{14}, v_{15}, v_2, v_{02})$.

Step 3: (b1). If point P is not in one of the rectangles, or in the central polygon, point P is *outside* the area and the computation is terminated.

Step 4: (b2). If point P is in neither of the rectangles, but in the central polygon, point P is *inside* the area and the computation is terminated.

Step 5: (b3). If point P is in one of the rectangles, say in rectangle q_{11}, go down one level on the strip tree, and repeat the above procedure until reaching the end of the tree branches.

Using the above procedures, we can compute point enclosure in a curve with thousands of points in only a dozen or so steps, depending on where the point is located. If the point lies outside of the area and away from the border, it may take only one cycle of computation for the enclosure test. If the point lies inside and close to the center of the area, the enclosure test may also take only one cycle of computation. When a point is located in one of the rectangles, the enclosure computation may possibly terminate in a few cycles. The worst case occurs only when a point happens to be located so that it causes the enclosure computation to search down to the bottom of a tree branch.

Chapter 3

Point-Polyhedron Enclosure Detection

Point-polyhedron enclosure detection, the problem of determining whether a point lies within a polyhedron in 3D space, occurs frequently in applications such as computer vision, pattern recognition, computer-aided design, and other areas involving computational geometry [4, 7, 13, 22], especially in object interference detection in a solid modeling system. Object interference detection is a procedure for investigating geometric interference between parts to avoid physically unrealizable situations with objects occupying the same region of space and to assure that parts fit correctly and mechanisms move correctly. Detections for interference are done by simulating the objects in their desired positions and testing for various types of interference or performing dynamic interference checking to determine whether an assembly is physically realizable.

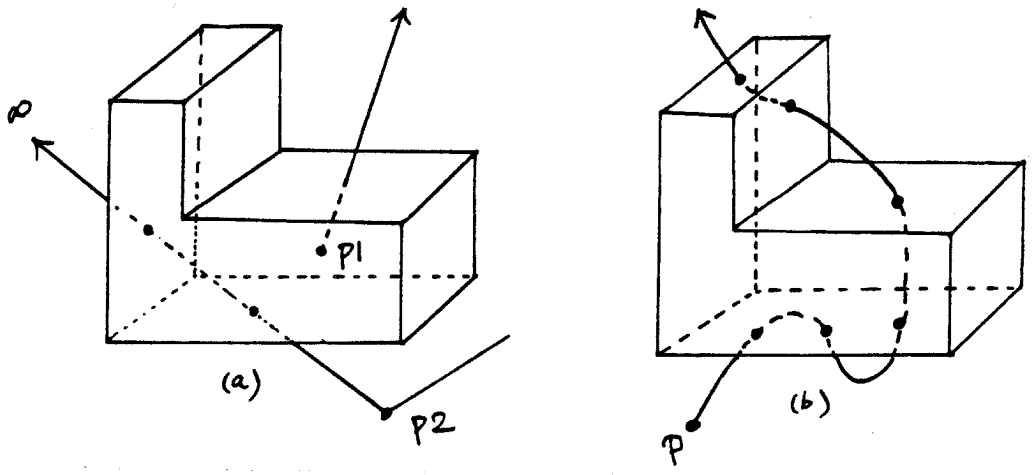

Figure 15
(a) Point P_1 is located inside the polyhedron; point P_2 is located outside the polyhedron. (b) A curve testing vector.

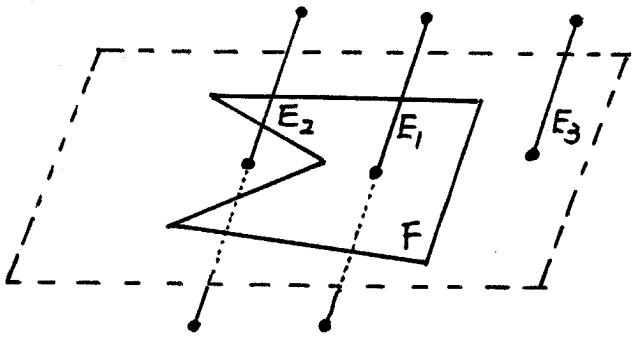

Figure 16
Edge E_1 penetrates face F. Both ends of edge E_2 are on opposite sides of F, but E_2 does not penetrate face F, Both ends of edge E_3 are on same side of F.

The conventional method for solving point-polyhedron enclosure problem involves determining the parity of intersections between the polyhedron and a testing vector that extends from the point under question to infinity [16]. If the vector makes an odd number of intersections with the polyhedron, then the point is enclosed by the polyhedron; if even number of intersections, then the point is outside the polyhedron (see Figure 15a). This is also true even when the testing vector is a curve instead of a straight half line (see Figure 15b).

The total number of intersections between a vector and a polyhedron can be determined by examining each face of the polyhedron to discover whether it is penetrated by the testing vector. A face is possibly penetrated by a vector only when each end of the vector is located on opposite sides of the face. If this is so, We then must check whether the intersection point of the testing vector with the plane containing the face is enclosed by the face. If both these conditions hold true, then we know that the face is penetrated by the vector, as E_1 shown in Figure 16. If both ends are on opposite sides of the face but the intersection point is not enclosed by the face, then the vector does not penetrate the face, as E_2 shown in Figure 16. If both ends are

on the same side of the face, as E_3 shown in Figure 16, the vector simply does not penetrate the face. This 3D point enclosure problem is thus reduced into a series of 2D point-polygon enclosure detections. The number of 2D detections required depends on the number of possible intersections between the polyhedron and the testing vector.

3.1 Point-polyhedron enclosure detection

However, it is important to notice that if a point is inside a polyhedron, no matter which direction a testing vector points toward, the vector always penetrates the closest face from inside to outside. If a point is outside a polyhedron, the testing vector either has no intersection with the polyhedron, or penetrates the closest face from outside to inside. We can, therefore, simply the problem by searching for the face which is penetrated by the testing vector and is closest to the point and determining the phase of the face in which the testing point lies. We say that a point is in the positive phase of a closest face if the testing vector penetrates the face from inside to outside; a point is in the negative phase of a closest face if the testing vector penetrates the face from outside to inside.

Based on this observation, we can further improve the efficiency of the conventional method by searching for only the face that is penetrated by a testing vector and is closest to the testing point; and therefore, we need not to perform a 2D enclosure detection on every possibly penetrated face. A face would require a 2D enclosure detection only if it makes a closer intersection with the testing vector than the one successfully examined previously. Those faces which may be penetrated by the vector but have farther intersections require no 2D enclosure detection. By this means, we can reduce the number of 2D point-polygon enclosure detections required during the entire operation.

Ideally, this method has the potential for drastically reducing the amount of work necessary in point-polyhedron enclosure detection. The amount of work in this algorithm, therefore, depends on how the faces of the polyhedron are traversed. On the average, the algorithm takes about half as much work as the conventional method. In the best case, the closest face is caught as the first try, we then need only one 2D

enclosure detection. However, if the faces are traversed from the farthest one to the closest one, then it takes the same amount of work as the conventional method does. The chances of this happening, however, are very small.

3.2 Point-polyhedron enclosure detection algorithm

Following is an algorithm for point-polyhedron enclosure detection. We first introduce some notation used in the algorithm. A simple polyhedron in 3D space is composed of a set of v vertices $(v_1, v_2, .., v_v)$, a set of e edges $(E_1, E_2, .., E_e)$, and a set of f faces $(F_1, F_2, .., F_f)$. A face is a simple planar polygon which can be either convex or concave. A face is described by a sequence of vertices $v_i =< x_i, y_i, z_i >$ and edges $E_i = (v_i, v_j)$. Two adjacent faces intersect by an edge and each edge connects two vertices. A vertex is possibly connected to more than three faces. Each face F_i is described by a plane equation $a_i x + b_i y + c_i z + d_i = 0$, and a sequence of vertices $< v_i^1, v_i^2, .., v_i^m >$ surrounding the normal vector in a clockwise direction, where $< a_i, b_i, c_i >$ is the outward unit normal vector.

Within the 3D point-enclosure detection algorithm below, a flag FACE points to the current minimum-distance face. A flag CID (current intersection distance) keeps track of the minimum distance of the penetrated faces which have been examined. A flag STATE which takes six scalars (normal, V-S, E-S, V-V-S, V-E-S, or E-E-S) indicates the current state of intersection, which could be either a normal intersection or a singularity. It it is a singularity, flags v_o, v_1, E_o, E_1 indicate the coincident elements. For an efficient computation, we always draw a testing vector toward the positive z-axis direction. By doing so, we can simplify the calculation of the distance from the testing point to the intersection of the testing vector with a face. Let us denote the coordinates of the testing point by $P =< x_o, y_o, z_o >$.

This algorithm works by searching among the faces of a polyhedron for the closest face which is penetrated by a testing vector. A distance of '−99' is stored in a flag 'CID', initially, for indicating that there is no intersection yet. The algorithm then looks for the first face which is penetrated by the testing vector, and then stores the intersection distance in 'CID'. The distance in CID is then used as a threshold

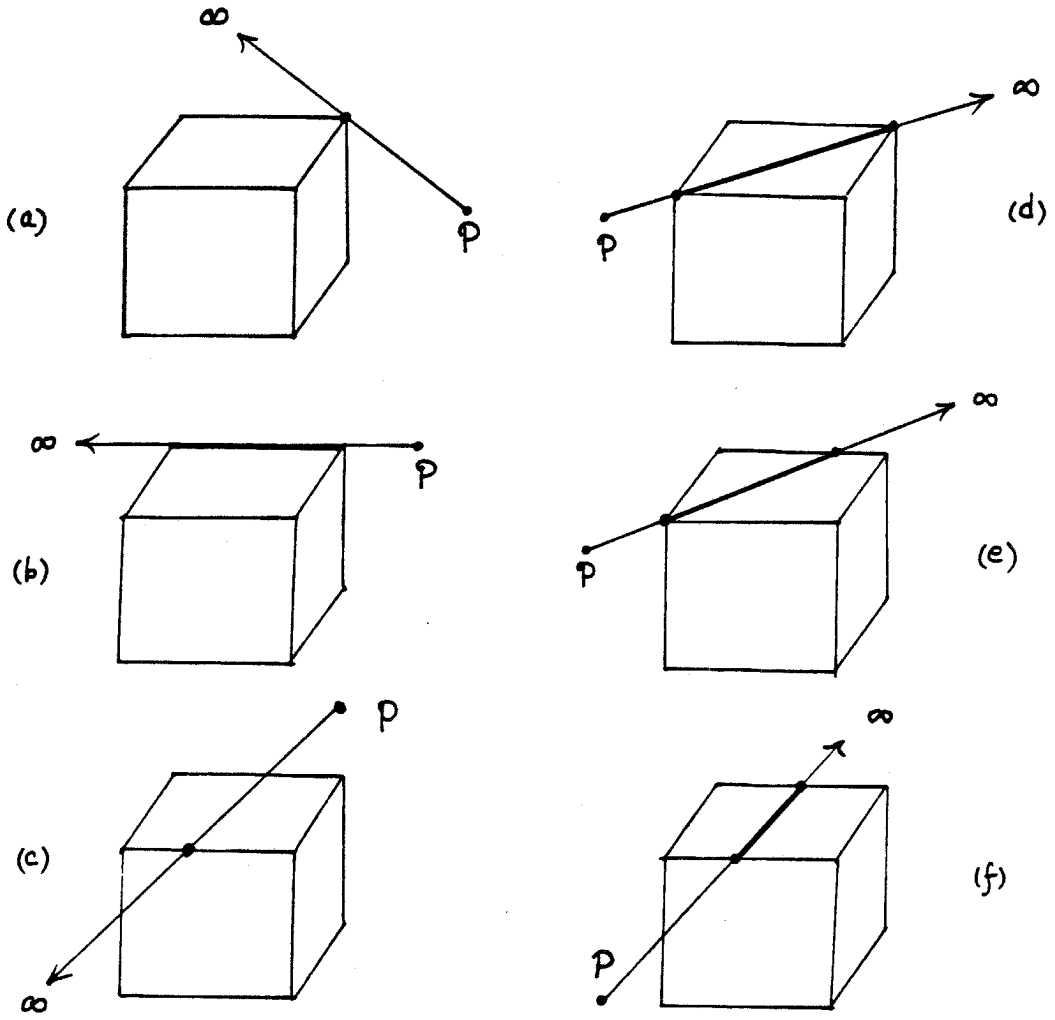

Figure 17

Singularity encountered in a 3D point enclosure detection.

to discard those faces having farther intersections. Only when a face has a closer intersection would it require a 2D point-polygon enclosure detection. If a face is successfully examined, its intersection distance then replaces the one stored in 'CID'. This distance is again used as a new threshold for examining next coming faces. By doing this recursively, the closest penetrated face can be eventually found.

Singularities are situations where a testing vector coincides with a vertex, an edge, or a face of the polyhedron, Figure 17. There are all kinds of singularities capable

of occurring in a 3D point enclosure detection. We will discuss methods for solving all these singularities in the next chapter; here we simply assume these routines are available. P is a testing point and Q is the polyhedron in operation.

(1) 2D-point-polygon-enclosure-detection(P, Q) routine:

detecting enclosure of P by polygon Q.

(2) V-singularity(P, v_o) routine:

v_o is the coincident vertex.

(3) E-singularity(P, E_o) routine:

E_o is a coincident edge.

(4) V-V-singularity(P, v_o, v_1) routine:

v_o and v_1 are coincident vertices.

(5) V-E-singularity(P, v_o, E_o) routine:

v_o is a coincident vertex and E_o is a coincident edge.

(6) V-E-singularity(P, E_o, E_1) routine:

E_o and E_1 are coincident edges.

Below is the 3D point-enclosure detection algorithm:

(Step 1) [Initialize the face index and the CID flag]

Set $i \Leftarrow 1$; CID $\Leftarrow$ maximum; STATE $\Leftarrow$ normal; FACE $\Leftarrow 0$.

(Step 2) [Skip to the next face]

$i \Leftarrow i + 1$;

If i $>$ f (total number of faces) or CID=0 then go to step 9.

(Step 3) [Determining the possibility of penetration]

IF $c_i = 0$ then go to step 6;

$z \Leftarrow -(a_i x_o + b_i y_o + d_i)/c_i$;

$P' \Leftarrow < x_o, y_o, z >$;

If $z < 0$ or $z >= CID$ then go to step 2;

(Step 4) [2D point-polygon enclosure detection]

CALL 2D-point-polygon-enclosure-detection(P', F_i) routine;

If P' is outside polygon F_i then go to step 2;

If P' is on boundary of F_i then go to step 7;

(Step 5) [Update flag CID]

CID $\Leftarrow z$; FACE $\Leftarrow i$; STATE $\Leftarrow$ normal;

Go to step 2;

(Step 6)[A face parallel to the testing vector]

If $(a_i x_o + b_i y_o + c_i z_o + d_i) \neq 0$ then go to step 2;

(Step 7) [Resolving Singularities]

If V-singularity then set STATE $\Leftarrow$ V-S;

find coincident vertex v_o ; call V-singularity(P , v_o) routine;

else if E-singularity then set STATE $\Leftarrow$ E-S;

find coincident edge E_o ; call E-singularity(P , E_o) routine;

else if V-V-singularity then set STATE $\Leftarrow$ V-V-S;

find coincident vertices v_o and v_1 ; call V-V-singularity(P , v_o , v_1) routine;

else if V-E-singularity then set STATE $\Leftarrow$ V-E-S;

find coincident vertex v_o and edge E_o ; call V-E-singularity(P , v_o , E_o) routine;

else if E-E-singularity then set STATE $\Leftarrow$ E-E-S;

find coincident edges E_o and E_1 ; call V-E-singularity(P , E_o , E_1) routine;

$z \Leftarrow$ intersection distance;

If singularity is significant then CID $\Leftarrow z$; go to step 2;

(Step 8)[Determining the phase]

Call phase routine;

If phase=positive then the point P is inside; otherwise P is outside.

The intersection distance in step 7 is measured as the distance between the testing point and the intersection of the testing vector with the coincident element. The coincident element could be a vertex as in a V-singularity, a point as in an E-singularity, or two elements, a vertex and/or an edge, as in a complex singularity. In a complex singularity, the intersection distance means the closer distance of the two intersections with the two coincident elements.

The PHASE routine called up in the last step determines in which phase the testing point P lies. If the testing point is in the positive phase of the closest face, P

is contained in the polyhedron; otherwise, P is outside. The phase of a face in which a testing point P lies can be determined by the sign of the plane equation value by substituting the coordinates of the point into the the plane equation of the face.

3.3 Phase routine

Generally, the closest element found in a point-enclosure-detection routine is a face. In that case it is very easy to determine the phase by calculating the plane-equation value of P with respect to the face. The sign of the value determines the phase of the testing point. However, there are cases where the closest element is in a singular situation; and this complicates the determination of the phase. We must, therefore, know how to determine the phase in a singularity case, even though the possibility of encountering a situation like this is very low.

If the closest element is an E-singularity, the phase can be determined by the region pattern bits of P with respect to the two consecutive faces F_0 and F_1 of the coincident edge E_0. The phase is positive if the testing point is in the $(++)$ region of F_0 and F_1; the phase is negative if P is in the $(--)$ region instead. Notice that we do not consider the cases where P is in $(+-)$ or $(-+)$ regions since only when P is in $(++)$ or $(--)$ regions can it be a significant singularity.

If the closest element is a V-singularity, the face that determines the phase is one of the converging faces of the coincident vertex closest to the testing vector.

We know that an E-E-singularity is significant when one of its coincident edges is convex and another is concave. If both coincident edges are convex or concave, the E-E-singularity is trivial and is neglected. The phase in an E-E-singularity can thus be determined by the edge with a closer intersection to the testing point. If the closer edge is concave and the farther one is convex, as in Figure 18a, the phase is positive. Conversely, if the closer one is convex then the phase is negative, as in Figure 18b.

For a V-E-singularity with coincident vertex v_0 and edge E_0 , the phase can be determined by edge E_0. If edge E_0 has a closer intersection to the testing point than v_0 and if E_0 is concave, the phase is positive, as in Figure 19a; otherwise, the phase is negative if E_0 is convex, as in Figure 19b. However, if v_0 is closer to P than

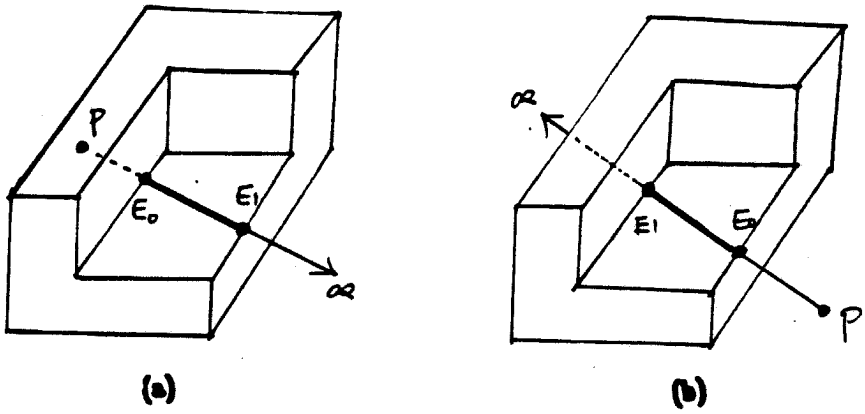

Figure 18

An E-E-singularity where (a) the closer edge E_o is concave and the farther edge E_1 is convex, (b) the closer edge E_o is convex and the farther edge E_1 is concave.

the edge E_o and if E_o is convex, the phase is positive , as in Figure 19c; otherwise, it is negative if E_o is concave, as in Figure 19d.

A V-V-singularity is significant only when one of its local V-singularities is significant and another is trivial. Let CV and CC represent the number of convex and concave transitions, respectively. From the singularity criterion, we know that a significant V-singularity has $CC = CV$ with its testing vector traveling from the interior to the exterior or vice versa. A trivial V-singularity has either $CC = CV + 2$ if the vector travels from interior to interior or $CV = CC + 2$ if the vector travels from exterior to exterior. Assume that within the V-V-singularity, v_o is significant and v_1 is trivial. The phase in this case can be determined from the trivial singularity at v_1.

If v_o is closer to the testing point than v_1, the phase is positive if v_1 has $CV = CC + 2$, as in Figure 20a; and it is negative if v_1 has $CC = CV + 2$, as in Figure 20b. The phase in Figure 20a is positive because the testing vector travels from the interior to exterior on v_o and from exterior to exterior on v_1. The testing point is thus inside the polyhedron. We can also tell that v_1 has $CV = CC + 2$ since the testing vector travels from exterior to exterior on v_1. The phase in Figure 20b is negative because

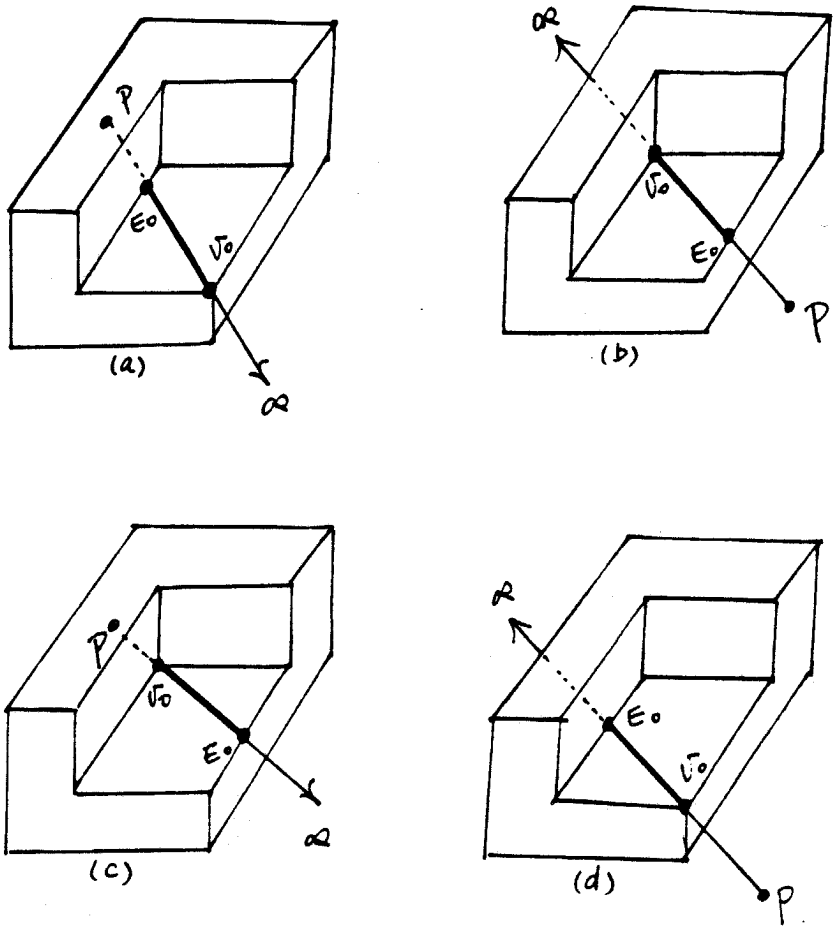

Figure 19

A V-E-singularity where (a) edge E_o is closer to the testing point and concave, (b) edge E_o is closer to the testing point and convex, (c) edge E_o is farther and convex, (d) the coincident edge E_o is concave.

the testing vector travels from exterior to interior on v_o and from interior to interior on v_1. The testing point is in the interior of the polyhedron. On the other hand, if v_1 is closer to the testing point than v_o, the phase is positive if v_1 has $CC = CV + 2$, as in Figure 20c; it is negative if v_1 has $CV = CC + 2$, as in Figure 20d.

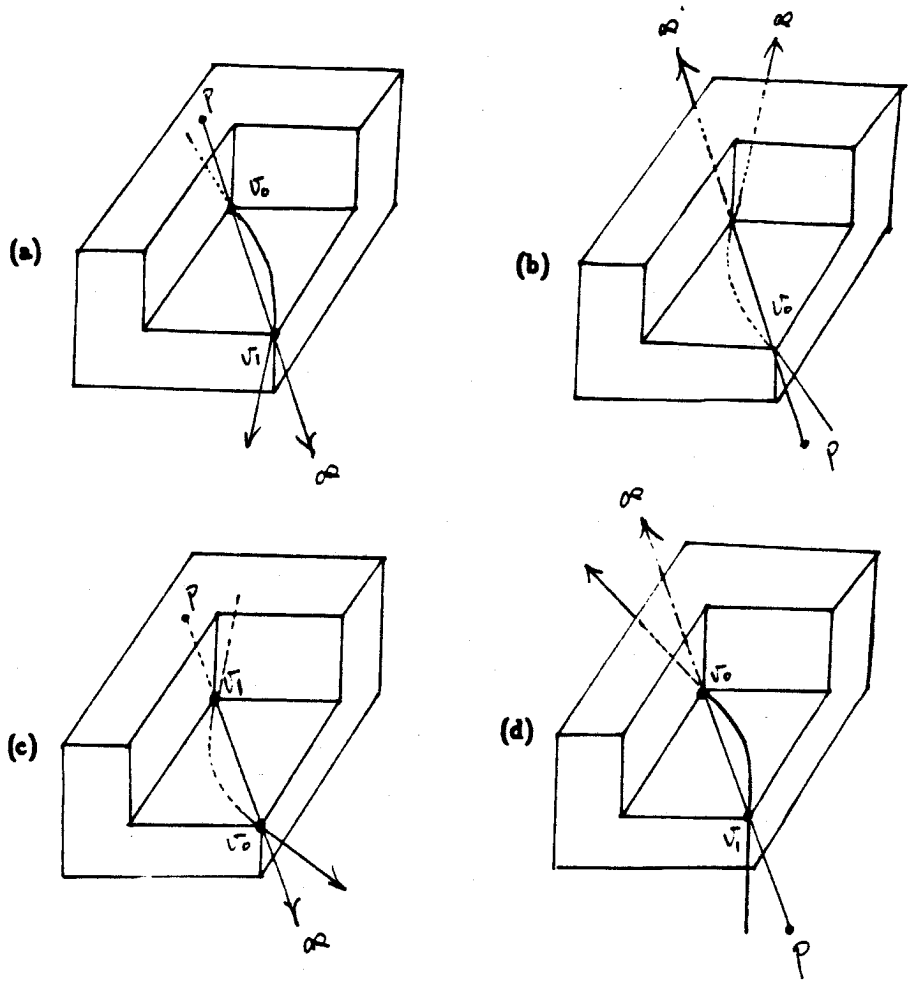

Figure 20

A V-V-singularity where the testing vector travels (a) from interior to exterior on v_0, and from exterior to exterior on v_1, (b) from exterior to interior on v_0, and from interior to interior on v_1, (c) from interior to interior on v_1, and from interior to exterior on v_0, (d) from exterior to exterior on v_1, and from exterior to interior on v_0.

Chapter 4

Resolving Singularities in Point-Polyhedron Enclosure Detection

Just as with point-polygon-enclosure detection, the problem of determining the inclusion of a point in a polyhedron is encountered frequently in many applications, [4, 6, 7, 13, 22]. However, an extra degree of freedom makes the 3D point-polyhedron containment analysis much more difficult than the 2D problem, mainly because of the geometrical increase in the number of possibly penetrated faces, and the lack of an efficient and accurate means for testing them. Resolution of 2D singularities based on local data alone was made possible due to the sequential nature of the ordering of the polygon elements (edges and vertices). In 3D space, however, there is no particular sequence in which the faces of the polyhedron are ordered.

Establishing the containment relationship between a point and a polyhedron can be accomplished, as in the 2D case, by counting the parity of intersections between the polyhedron and a vector extending from the point to infinity. However, the parity count can be impaired by singularities which arise when a testing vector coincides with a vertex, an edge, or a face of the polyhedron. If we misjudge the significance of the singularity, we can derive an incorrect parity count, and thus turn the point-polyhedron containment relationship inside-out or vice versa. Singularities resolve into two situations where (1) a testing vector penetrates a polyhedron from its interior to its exterior or vice versa, or where (2) a testing vector is tangent to a polyhedron and causes no inside/outside transition. The former case is a significant intersection and can be counted for parity, while the latter case is a trivial one and cannot be counted for parity.

Singularities can be resolved by either avoiding them or solving them directly. One way to avoid singularities is to choose a different testing vector that causes no

singularity. However, because a certain amount of trial-and-error is involved in finding a singularity-free vector, trying to avoid a singularity can actually be more expensive than a direct solution. Kalay [16] resolves a singularity by removing the elements which cause the singularity (the coincident edge and vertex and all the edges which converge on it) and then testing the contour of the merged faces. However, this method is not applicable to cases where a testing vector is coplanar with a face of a polyhedron.

We propose a singularity criterion in this chapter which we believe leads to a simpler, superior solution to all singularities encountered in point-polyhedron enclosure detection. We first categorize all the singularities into two basic types and four complex types. We then show that a complex singularity can be decomposed into two basic singularities by bending or perturbing a testing vector; the two components can then be resolved and their results combined.

4.1 Categorizing singularities

The singularities encountered in a point-polyhedron enclosure detection fall into the following six categories:

- The testing vector intersects a vertex of the polyhedron but does not lie in any of its connecting planes, as in Figure 21a.
- The testing vector intersects an edge of the polyhedron by a point, as in Figure 21b.
- The testing vector intersects the polyhedron by an edge, i.e., the testing vector lies at the intersection of two faces of the polyhedron, as in Figure 21c.
- The testing vector lies on a face of the polyhedron and intersects two vertices of that face, as in Figure 21d.
- The testing vector lies on a face of the polyhedron and intersects a vertex and an edge of that face, as in Figure 21e.
- The testing vector lies on a face of the polyhedron and intersects two edges of that face, as in Figure 21f.

The first two categories describe basic singularities and the others describe complex singularities. The first basic singularity is called a *V-singularity*. A V-singularity is a case where a testing vector intersects a vertex of a polyhedron but

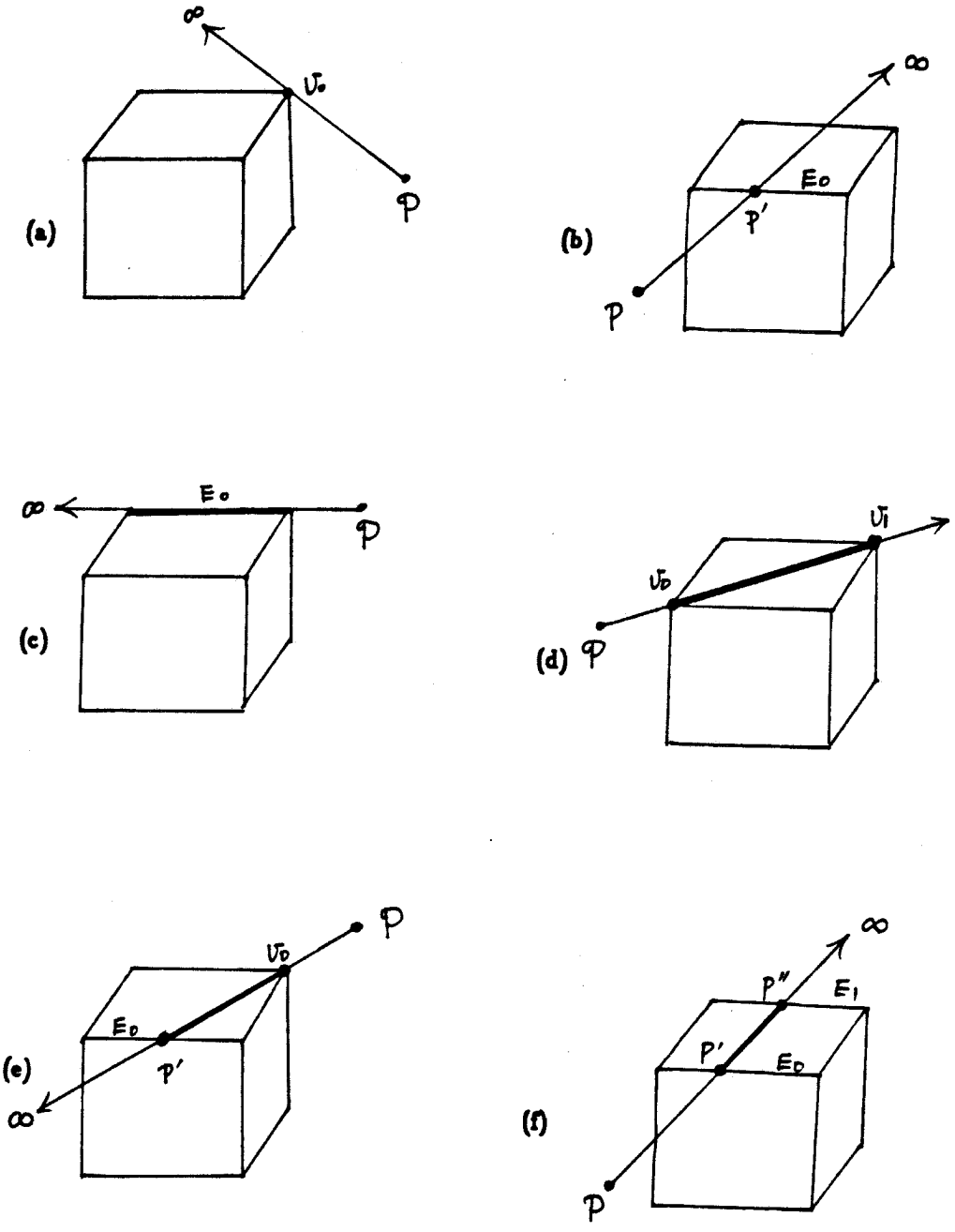

Figure 21

The testing vector intersects a polyhedron by (a) a vertex, (b) a point on an edge, or (c) the vector is collinear with an edge, or (d) intersects by two vertices of a face, (e) a vertex and an edge, or (f) two edges of a face.

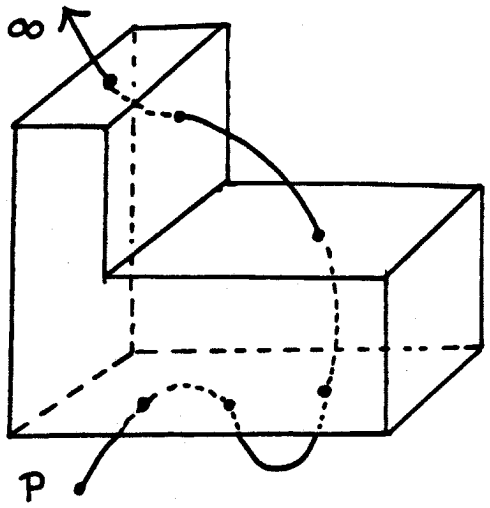

Figure 22

A bent testing vector will not disturb the parity count as long as it starts from the testing point and extends toward infinity.

is not coplanar with any of its converging planes, as Figure 21a shows. The second basic singularity type, the *E-singularity*, is a case where a testing vector intersects an edge of the polyhedron but is not collinear with the edge, as Figure 21b shows.

As we know, perturbing a testing vector does not disturb the parity of the intersections (see Figure 22) as long as the vector starts from the testing point and stretches toward infinity. In the following figures we show how a complex singularity can be decomposed into a combination of V- and/or E-singularities by bending the testing vector. We later discuss how a complex singularity can be resolved by resolving the two components and combining their results.

The singularity in Figure 21c can be decomposed into two local V-singularities by pinning down both ends of the vector at vertices v_0 and v_1 and detaching the vector from the coincident edge E_0 . The vector was originally collinear with E_0 ; but it now intersects the polyhedron by only two vertices, as Figure 23a shows. We call this a *F-F-singularity.*

The singularity in Figure 21d can also be decomposed into two local V-singularities by pinning down the vector at v_0 and v_1 and detaching the vector from the

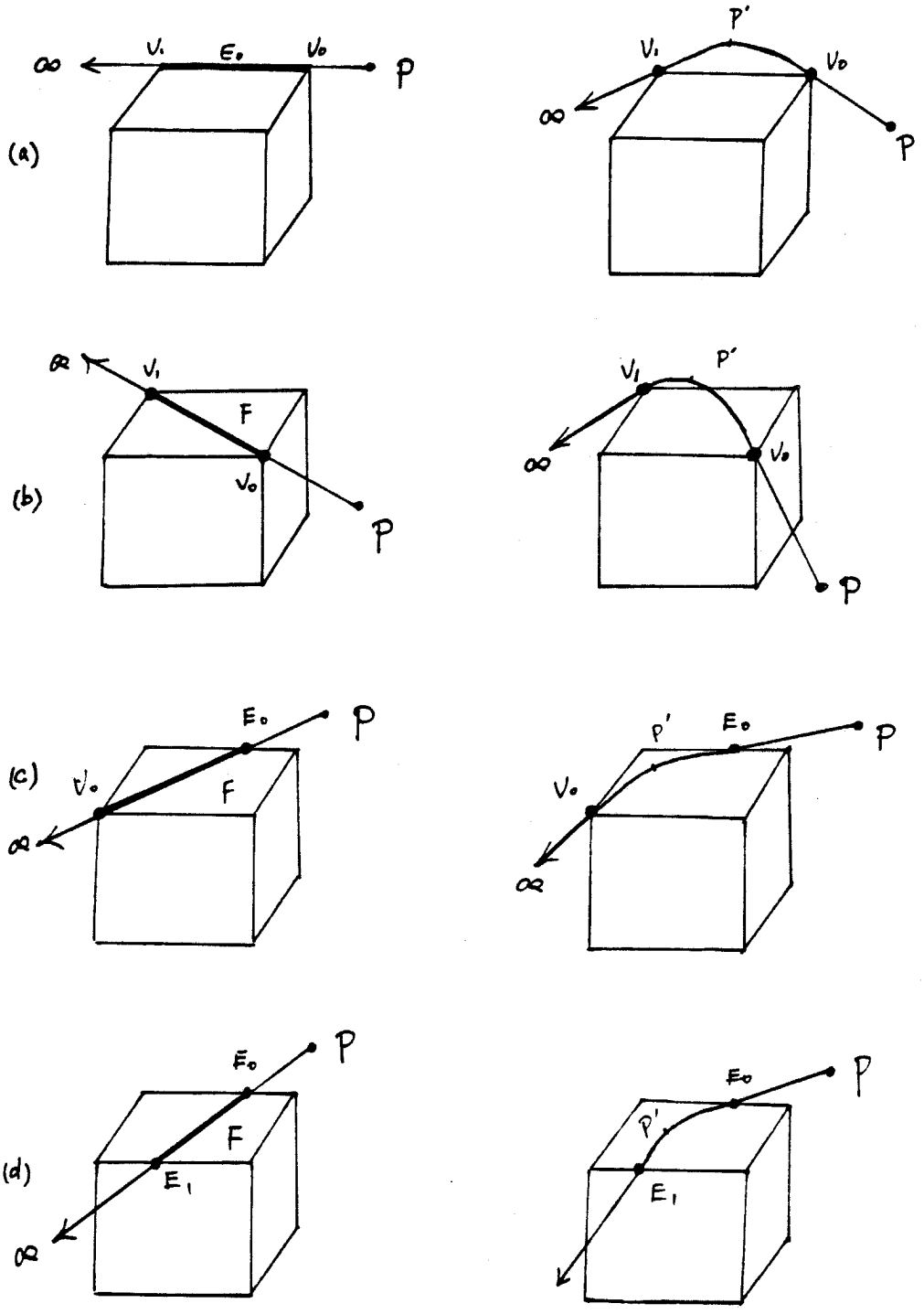

Figure 23

The vector is deformed to intersect the polyhedron by (a) (b) two vertices v_o and v_1, (c) a vertex v_o and an edge E_o, or (d) by two edges E_o and E_1.

coincident face F . The vector was originally coplanar with the face F , but now it intersects the polyhedron only by v_o and v_1 , as Figure 23b shows. This is called a *V-V-singularity.*

The singularity in Figure 21e is decomposed into a V-singularity and an E-singularity by bending the vector so that it intersects only a vertex and an edge of the polyhedron, as in Figure 23c. We call this a *V-E-singularity* with a local V-singularity and a local E-singularity.

The singularity in Figure 21f is decomposed into two E-singularities by bending the vector so that it intersects two edges of the polyhedron, as in Figure 23d. We call this an *E-E-singularity* with two local E-singularities.

4.2 Resolving a V-singularity

The method for resolving a V-singularity is based on a singularity criterion proposed later as an extension of the transition criterion proposed in the second chapter.

A 3D polyhedron W can be described by a set of vertices $V(W) = (v_1, v_2, .., v_v)$, a set of edges $E(W) = (E_1, E_2, .., E_e)$, and a set of faces $F(W) = (F_1, F_2, .., F_f)$. A face is a planar polygon that can be either convex or concave. A face is defined by a sequence of coplanar vertices, where each two consecutive vertices constitute an edge and two consecutive faces are intersected by an edge. A vertex may connect more than three faces.

To determine the containment relationship between a testing point P and a polyhedron W, a testing vector is drawn from P to infinity, and the parity of intersections between the testing vector and the polyhedron is determined. Assume that the testing vector causes a singularity by coinciding with a vertex v_o of the polyhedron. Let $(F_1, F_2, .., F_m)$ represent the neighboring faces of v_o which surround v_o in a sequential order. Let $(E_1, E_2, .., E_m)$ represent the neighboring edges of v_o which connect pairs of consecutive faces in $(F_1, F_2, .., F_m)$, i.e., F_{i-1} and F_i intersect E_i, as in Figure 24. We also assume that the testing vector is not coplanar with any of the m faces.

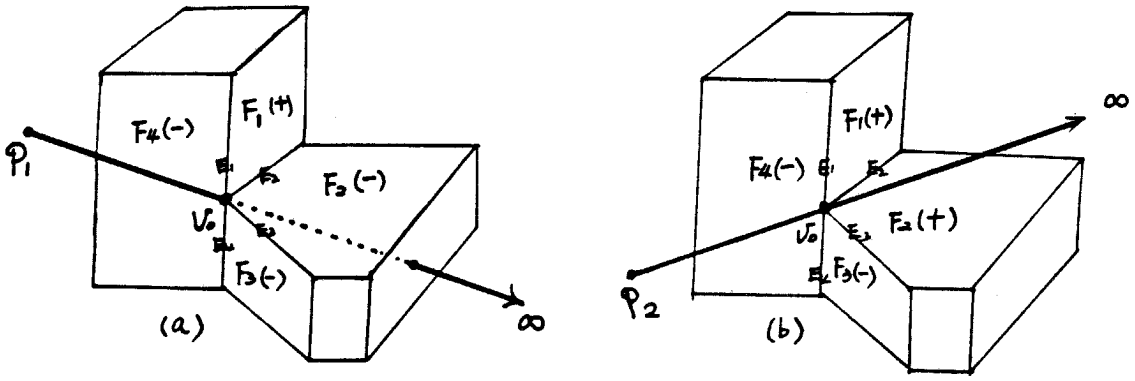

Figure 24

E_1 and E_3 are convex edges. E_2 and E_4 are concave edges. (a) The region pattern of P_1 with respect to the faces is (+———). (b) The region pattern of P_2 with respect to the faces is (++——).

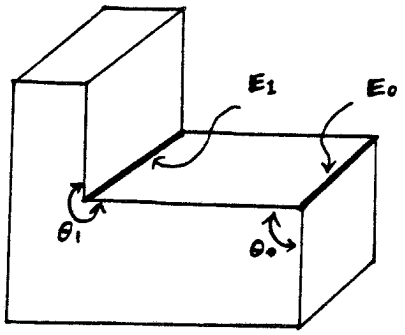

Figure 25

Angle θ_o is less than π, and edge E_o is convex; angle θ_1 is greater than π, and edge E_1 is concave.

We define an *edge pattern* of a polyhedron in terms of the properties of the polyhedron's edges. This edge pattern corresponds to a vertex pattern of a polygon.

An edge is *concave* if the angle between the two adjacent faces covering the interior of the polyhedron is reflex; otherwise it is *convex*. For instance, edge E_o in Figure 25 is convex, but edge E_1 is concave. We define the edge pattern, $EP(W)$, on a subset of edges of a polyhedron W as

$$\begin{aligned} EP(E_1, E_2, .., E_m/W) &= (EP_1, EP_2, ..., EP_m), \text{ where} \\ EP_i &= 1 \quad \text{if } E_i \text{ is convex, or} \\ &= -1 \quad \text{if } E_i \text{ is concave.} \end{aligned} \tag{4.1}$$

Let G_i be the plane containing face F_i and described by plane equation $G_i(x, y, z) = 0$. This plane divides space into two regions: a positive region satisfying $G_i(x, y, z) > 0$; and a negative region satisfying $G_i(x, y, z) < 0$. We always define a plane equation such that the interior of a polyhedron always lies in the positive region of a face. We define a ***region pattern*** of a point $P =< x, y, z >$ with respect to a subset of faces of a polyhedron as

$$\begin{aligned} RP(P, F_1, F_2, .., F_m) &= (RP_1, RP_2, ..., RP_m), \text{ where} \\ RP_i &= + \quad \text{if } G_i(x, y, z) > 0, \text{ or} \\ RP_i &= - \quad \text{if } G_i(x, y, z) < 0. \end{aligned} \tag{4.2}$$

A bit RP_i is positive if P lies in the positive region of F_i, and is negative if P is in the negative region. Here the positive or negative regions of a face refer to the positive or negative regions divided by the plane containing the face.

We then define a ***transition pattern*** (TP) of P with respect to a set of converging faces of a coincident vertex, $(F_1, F_2, .., F_m)$, based on the region pattern defined above:

$$\begin{aligned} TP(P, F_1, F_2, .., F_m) &= (T_1, T_2, ..., T_m), \text{ where} \\ T_i &= 0, \quad \text{if } RP_{i-1} = RP_i, \\ &= 1, \quad \text{if } RP_{i-1} \neq RP_i \text{ and } E_i \text{ is convex, and} \\ &= -1, \quad \text{if } RP_{i-1} \neq RP_i \text{ and } E_i \text{ is concave.} \end{aligned} \tag{4.3}$$

Since faces F_{i-1} and F_i are intersected by edge E_i , we say that P has a transition at E_i if $RP_{i-1} \neq RP_i$. Furthermore, if E_i is convex we say that P has a *convex transition*; if E_i is concave, we say that P has a *concave transition*. We can then count

the total transitions of P on $(F_1, F_2, .., F_m)$ by means of the following equation:

$$\text{transitions } (P/F_1, F_2, .., F_m) = \sum_{i=1}^{m} T_i \tag{4.4}$$

This leads to the following Singularity Criterion:

I. *We say that a testing vector has a significant intersection with a polyhedron at a testing point if the vector penetrates the surface of the polyhedron at that point. If a testing vector has a significant intersection with the polyhedron, it has an equal number of convex and concave transitions at the testing point, i.e.,*

$$\textit{transitions } (P/F_1, F_2, .., F_m) = 0. \tag{4.5}$$

II. *We say that a testing vector has a trivial intersection with a polyhedron at a testing point if the vector is tangent to the surface of the polyhedron at that point. If a testing vector forms a trivial intersection with the polyhedron, it has at the testing point a total of two more convex than concave transitions, or vice versa, i.e.,*

$$\textit{transitions } (P/F_1, F_2, .., F_m) = \pm 2. \tag{4.6}$$

Using this criterion, we can then determine whether a singularity is significant or trivial. The proof of the singularity criterion appears in the last section.

The following two examples in Figure 24 show how a singularity can be solved by using the singularity criterion. Points P_1 and P_2 are two testing points against the polyhedron W. A testing vector, drawn from P_1 to infinity, intersects W by vertex v_o and causes a singularity at v_o , Figure 24a, vertex v_o has four neighboring faces, F_1, F_2, F_3, and F_4. The intersections of F_1 with F_4 and F_2 with F_3 are convex edges E_1 and E_3, respectively, and the intersections of F_1 with F_2 and F_3 with F_4 are concave edges E_2 and E_4, respectively. The polyhedron W thus has an edge pattern defined on the four faces: $EP(F_1, F_2, F_3, F_4) = (1, -1, 1, -1)$.

P_1 is in the positive region of F_1 and the negative regions of F_2, F_3, and F_4, the region pattern of P_1 with respect to F_1, F_2, F_3, F_4 is

$$RP(P_1, F_1, F_2, F_3, F_4) = (+, -, -, -).$$

This means that P_1 has two transitions, one between F_4 and F_1, and the other between F_1 and F_2. Edge E_1 is convex and E_2 is concave. The transition pattern of point P_1 is, therefore, $TP(P_1, F_1, F_2, F_3, F_4/W) = (1, -1, 0, 0)$. In other words, P_1 has a convex transition at E_1 and a concave transition at E_2, and so it has a total of zero transitions. According to the singularity criterion, the testing vector penetrates the surface of W at vertex v_o .

Another testing vector is drawn from P_2 to infinity, as shown in Figure 24b; and this also causes a singularity at v_o . The region pattern of P_2 is

$$RP(P_2, F_1, F_2, F_3, F_4) = (+, +, -, -).$$

P_2 has a total of two convex transitions, one at E_1 between F_4 and F_1 and the other at E_3 between F_2 and F_3. According to the singularity criterion, the testing vector is tangent to the polyhedron at vertex v_o .

4.3 Resolving an E-singularity

Faces F_1 and F_2 in Figure 26a intersect a convex edge, and F_3 and F_4 in Figure 26b intersect a convex edge. Two planes G_1 and G_2 containing F_1 and F_2, respectively, separate the space under consideration into four regions $(++), (+-), (-+), (--)$ defined with respect to F_1 and F_2 as do two planes G_3 and G_4 containing F_3 and F_4, respectively. A testing vector in an E-singularity can only travel between opposite regions, e.g., from a $(++)$ to $(--)$ region or from $(+-)$ to $(-+)$ region; it can not travel between two neighbor regions such as from $(++)$ to $(+-)$ or from $(--)$ to $(-+)$. We can see from the figure that regardless the faces intersect a convex or a concave edge, a testing vector traveling between $(++)$ and $(--)$ regions always penetrates the surface of the polyhedron as Figures 26a1 and 26b1 show; likewise, a testing vector traveling from $(+-)$ to $(-+)$ or vice versa is always tangent to the surface of the polyhedron as Figure 26a2 and 26b2 show. Therefore, an E-singularity can be easily solved by calculating the region pattern of the testing point with respect to the two consecutive faces of the coincident edge. The singularity is significant if the testing point resides in $(++)$ or $(--)$ regions, otherwise it is trivial.

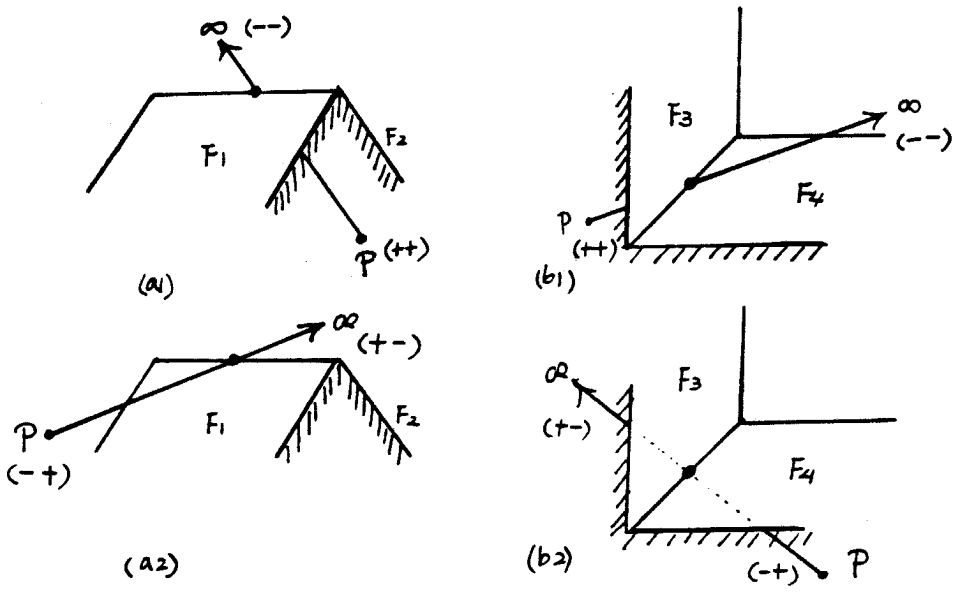

Figure 26

Faces F_1 and F_2 intersect a convex edge. Faces F_3 and F_4 intersect a concave edge. A testing vector traveling (a1) from (++) to (−−) region of F_1 and F_2; (b1) from (++) to (−−) region of F_3 and F_4; (a2) from (−+) to (+−) region of F_1 and F_2; (b2) from (−+) to (+−) region of F_3 and F_4.

4.4 Resolving complex singularities

A complex V-V-, V-E-, E-E-, or F-F-singularity is a combination of two local V- or E-singularities, and its global result is determined by combining results from its two local singularities. If both local singularities are significant, then the global result is trivial; since an inside/outside transition followed by an outside/inside transition, or vice versa, results in no global transition, as Figure 27a shows. If both local singularities are trivial, then the complex singularity is still trivial, Figure 27b. Only when one local singularity is significant and another is trivial is the complex singularity thus significant, as Figure 27c shows.

The V-V-singularity shown in Figure 21d can be simplified into two local V-

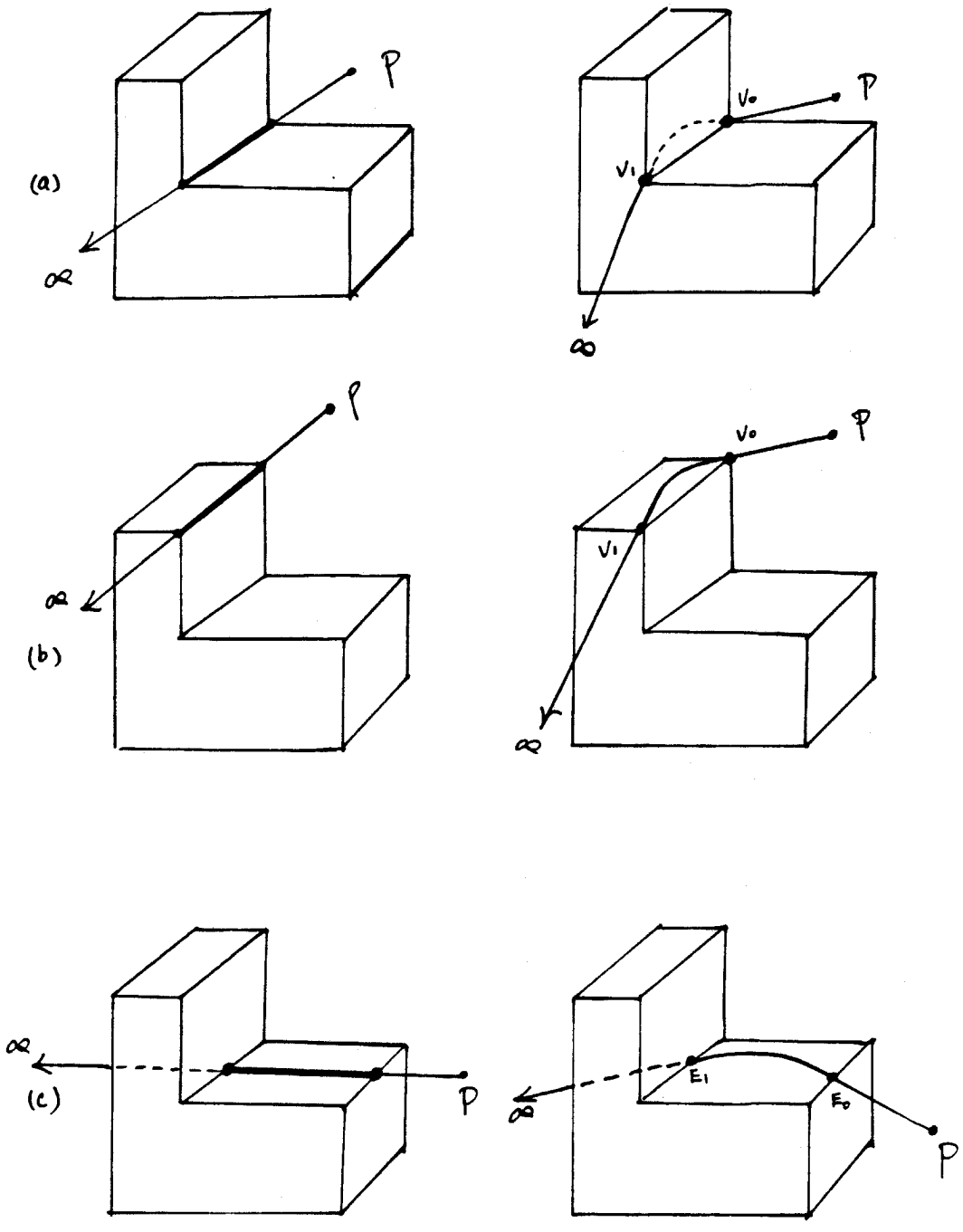

Figure 27

(a) The bent testing vector travels from outside to inside at v_o and then from inside to outside at v_1. (b) The bent vector has two trivial singularities at v_o and v_1, separately. (c) The singularity at E_o is trivial but the one at E_1 is significant. The result is thus a significant singularity.

singularities by detaching the testing vector away from the coincident face. This results the V-V-singularity into a combination of two V-singularities at vertices v_o and v_1 . Let $(F, F_2, .., F_m)$ and $(F, F_2', F_3'.., F_k')$ represent, the adjacent faces of v_o and v_1 , respectively. To calculate the region pattern of P with respect to $(F, F_2, .., F_m)$ and $(F, F_2', F_3'.., F_k')$, we notice that the region pattern bit of the testing point with respect to face F can not be defined because the vector is coplanar with F. By shifting the testing point P into the positive region of face F (and thus P' is moved into the negative region of F) as shown in Figure 23b, we can substituted the zero region pattern bit with (+). By applying the singularity criterion separately to v_o and v_1 with the region pattern bit with respect to F substituted by (+)in case of v_o and by (−) in v_1 , we can then determine the result of this V-V-singularity.

Figure 23a shows an F-F-singularity where the testing vector is collinear with an edge E_o =(v_o, v_1). Let F_1 and F_2 be the two adjacent faces of E_o described by plane equations $a_1x + b_1y + c_1z + d_1 = 0$ and $a_2x + b_2y + c_2z + d_2 = 0$, respectively. Let $(F_1, F_2, .., F_m)$ and $(F_1, F_2, F_3'.., F_k')$ represent the sequences of adjacent faces separately surrounding v_o and v_1. Notice that faces F_1 and F_2 appear in both sequences. To calculate the region pattern of P with respect to $(F_1, F_2, .., F_m)$ and $(F_1, F_2, F_3'.., F_k')$, we notice that two bits can not be defined (P with respect to faces F_1 and F_2), because the testing point $P =< x_o, y_o, z_o >$ is on both F_1 and F_2 , i.e., $a_1x_o + b_1y_o + c_1z_o + d_1 = 0$ and $a_2x_o + b_2y_o + c_2z_o + d_2 = 0$.

By bending the testing vector into one of the four regions separated by faces F_1 and F_2 , such as the (+, +) region, this F-F-singularity becomes a combination of two V-singularities at vertices v_o and v_1 as illustrated in Figure 23a. Thereafter, the two region pattern bits can be defined. We can then solve the original F-F-singularity by applying the singularity criterion separately to the two local cases. When calculating the region pattern of P with respect to $(F_1, F_2, .., F_m)$, if we choose to move point P into the (++) region, we substitute the two zero region pattern bits with (++) bits. When calculating the region pattern of P with respect to $(F_1, F_2, F_3'.., F_k')$, we substitute the two zero region pattern bits with (−−) instead. We can then determine whether this F-F-singularity is significant or trivial by combining the two local results

following the rules we explained above.

It is easy to see that the global result would stay the same if we chose different regions to bend the vector toward. If we had chosen to move point P into the $(+-)$ region, then when applying the criterion to v_o and v_1 , the two zero region pattern bits should be substituted by $(+-)$ in v_o and $(-+)$ in v_1 , respectively.

In case (e) we simplify the V-E-singularity into a V-singularity at vertex v_o and an E-singularity at edge E_o as Figure 23c shows. We bend the vector into the positive region of F so that we can substitute the region pattern bit with a $(+)$ bit and apply the criterion to solve the local V-singularity E-singularity.

In the last case (f) where a testing vector crosses two edges E_o and E_1 of the face F , the singularity is an E-E-singularity, Figure 23d. Again by bending the testing vector into the positive region of F , we can solve both local E-singularities by substituting the region pattern bit with $(+)$ bit when solving the E_o case and with $(-)$ bit when solving the E_1 case.

4.5 Application to 3D edge-face penetration detection

The singularity criterion described previously pays its great benefits when it is applied to the very time-consuming operation of edge-face-penetration detection. Conventionally, the object collision problem is solved by comparing all edges of one object against all faces of another object, and vice versa. This obviously may lead to hundreds of edge-face-penetration detections. Edge-face-penetrating detection traditionally skirts this danger by reducing the dimensionality of the problem, first by checking whether both ends of the edge are in opposite half-spaces divided by the extension plane of the face, and then by performing a 2D enclosure test to determine whether the intersection point of the edge with the extension plane is contained in the face. See Figure 28a and 28b for example.

However, it is possible to apply the singularity criterion to this problem directly without reducing its dimensionality. As shown in Figure 28c and 28d, by treating one end of the edge, p_1, as a coincident vertex, and the other end, p_2, as a testing point, we can set up a set of virtual convergent faces $(F_1, F_2, ..., F_n)$ and turn the edge-face-

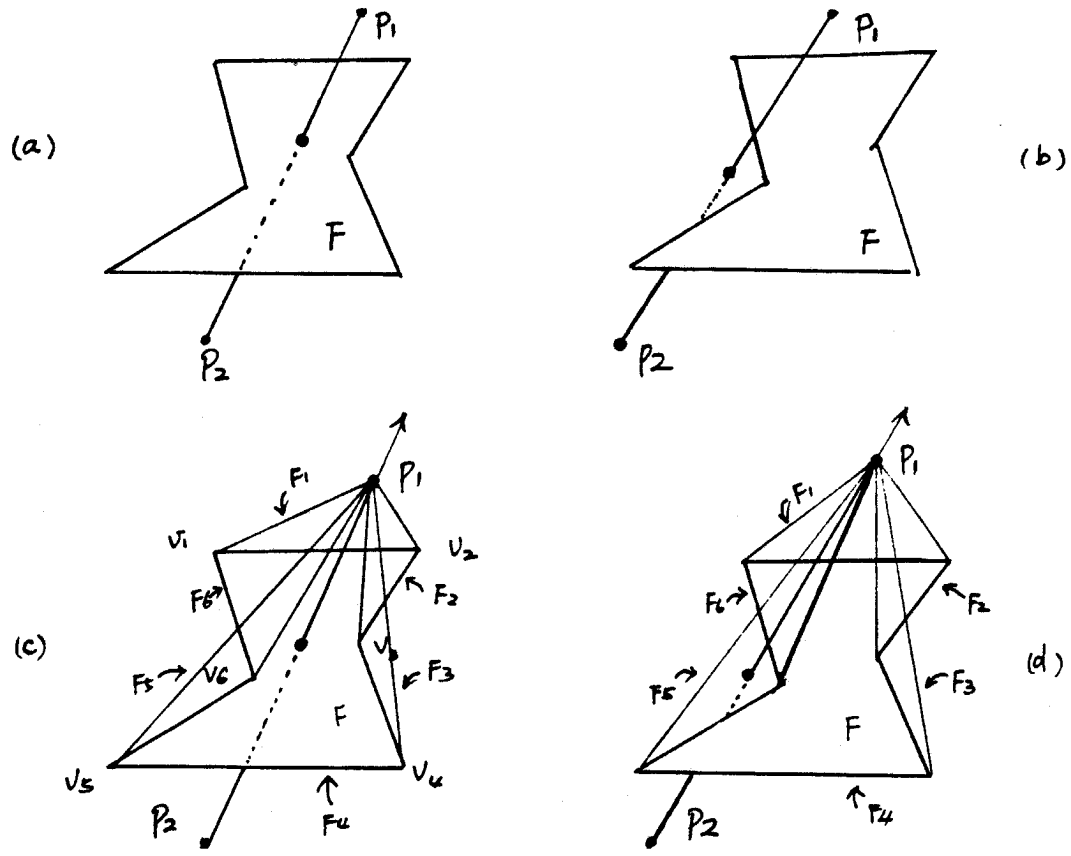

Figure 28

(a) Edge $E = (P_1, P_2)$ penetrates the face F. (b) Edge $E = (P_1, P_2)$ does not penetrate the face F. (c) (d) A set of virtual faces $(F_1, F_2, ..., F_6)$ is set up around P_1 which results in a V-singularity at P_1.

penetration problem into a V-type singularity. We can then solve the singularity by calculating the transition pattern of the testing point p_2 with respect to the faces $(F_1, F_2, ..., F_n)$. According to the singularity criterion, an edge penetrates a face only if the singularity is significant. Otherwise, if the singularity is trivial, we know that the edge does not penetrate the face.

Assume that face F is composed of vertices, $v_1, v_2, ..., v_n$. By treating p_1 as the

coincident vertex, a set of faces $(F_1, F_2, ..., F_n)$ is set up to constitute the converging faces of p_1. Each face F_i is specified by three points, (v_i, v_{i+1}, and p_1). The normal vector, $\mathbf{n}_i$, of face F_i is the outer product of vectors (p_1, v_i) with (p_1, v_{i+1}). The edge pattern bit of the edge E_i in between F_{i-1} and F_i can be determined by the inner product of vector (p_1, v_{i-1}) with the normal vector $\mathbf{n}_i$. The region pattern bit of the testing point p_2 with respect to a face F_i can be determined by the inner product of vector (p_1, p_2) with the normal vector $\mathbf{n}_i$.

According to the singularity criterion, if the transition pattern of the testing point has an equal number of convex and concave transitions, we know that edge E penetrates face F , as in Figure 28c, otherwise E does not penetrates face F , as in Figure 28d.

4.6 Proof of the singularity criterion

To analyze a V-singularity at vertex v_o , let us assume the existence of a sphere $S(r)$ centered at v_o , Figure 29a. The radius r of the sphere is so small that the sphere intersects the polyhedron only by those faces which converge to v_o . Let $(F_1, F_2, .., F_m)$ represent the set of converging faces of v_o . The intersection of the sphere $S(r)$ with $(F_1, F_2, .., F_m)$ creats a simple closed curve C , called the *trace of vertex* v_o on sphere S , as shown in Figure 29b. This trace is composed of as many sections as the number of faces in $(F_1, F_2, .., F_m)$, in which each section is an arc formed by the intersection of the sphere S with one of the converging faces. The sphere S is then separated into two areas by the trace C : (1) the interior which is merged in the polyhedron and (2) the exterior which is exposed in the air, Figure 29b.

Let $(G_1, G_2, .., G_m)$ represent the set of planes containing faces $(F_1, F_2, .., F_m)$, respectively. A plane G_i is described by a plane equation $G_i(x, y, z) = 0$. It is clear that all planes in $(G_1, G_2, .., G_m)$ intersect each other exactly at v_o . The volume of S is divided by $(G_1, G_2, .., G_m)$ into $2f$ regions: $(R_1, R^1, R_2, R^2, .., R_f, R^f)$. The total number of regions is dependent on how $(F_1, F_2, .., F_m)$ are located. Here we use (R_i and R^i) to denote a pair of dual regions, i.e., a testing vector entering S at region R_i and going through v_o would exit by its dual region R^i, and vice versa. If only

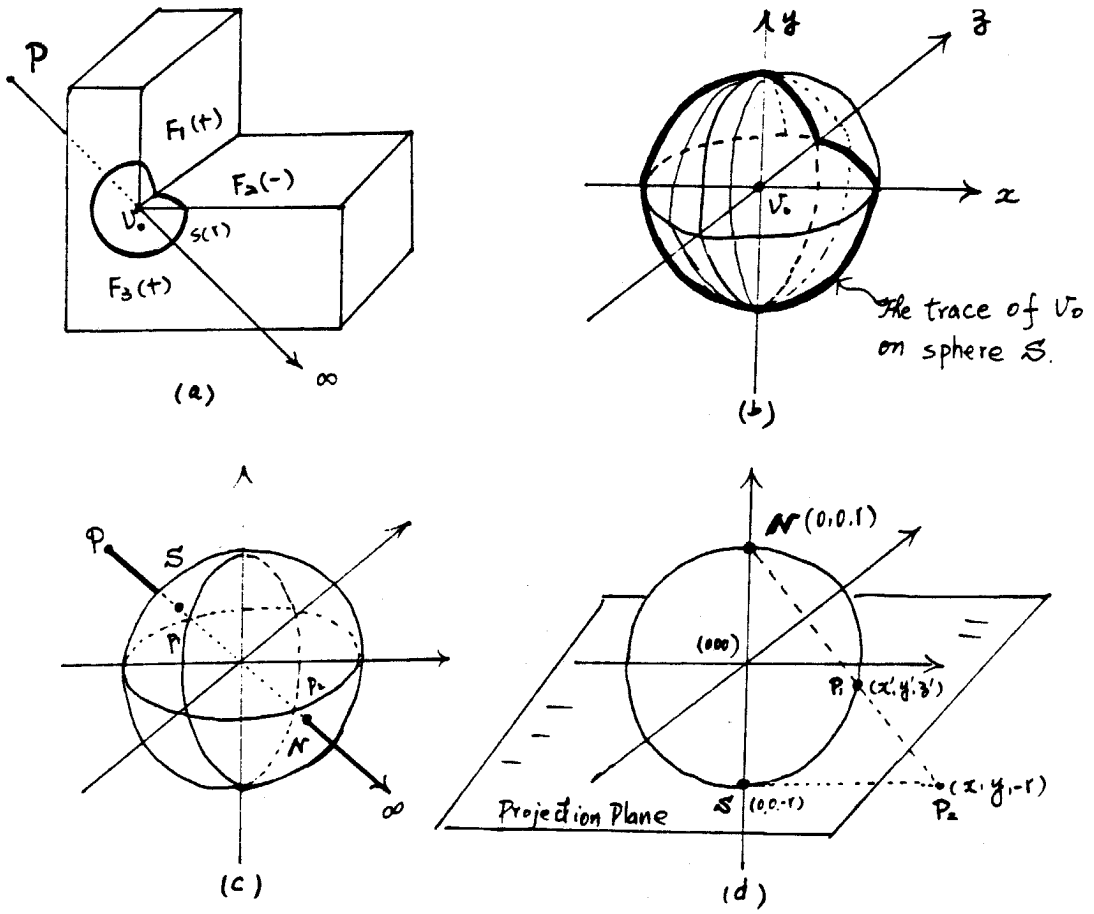

Figure 29

(a)(b) The intersection of the converging faces of v_o with a sphere $S(r)$ centered at v_o is the trace of v_o. (c) P_1 is the south pole and P_2 is the north pole of S. (d) Sphere S is projected from the north pole P_2 onto the projection plane by a one-to-one mapping, except the point P_2. The south pole P_1 is mapped to the origin of the projection plane.

$(F_1, F_2, .., F_m)$ are concerned, each region in $(R_1, R^1, R_2, R^2, .., R_f, R^f)$ defines a unique region pattern. A pair of dual regions has conjugate region patterns, since they are located at opposite sides of vertex v_o .

Two dual regions R_i and R^i can both lie either interior or exterior to the polyhedron, or one can reside in its interior and the other on its exterior. We also know that a testing vector entering at a region R_i must exit by its dual region. If a testing vector enters by an interior region of the polyhedron and exits by an exterior region, or vice versa, then it causes an inside/outside transition, i.e., the testing vector penetrates the surface of the polyhedron. In this case, we know that the testing vector has a significant intersection with the polyhedron. On the other hand, if the vector both enters and exits either on an interior or an exterior region, we know that the testing vector is tangent to the polyhedron. In this case, the testing vector has only a trivial intersection with the polyhedron. By categorizing (R_i and R^i) as a pair of ***significant regions*** if one of them resides in the interior and the other in the exterior and as a pair of ***trivial regions*** if they are both in the interior or exterior, we can easily determine whether a testing vector penetrates or is tangent to a polyhedron by determining whether it is traveling between a pair of significant or trivial regions.

Next we will prove the singularity criterion. We will show that if a testing vector travels between a pair of significant regions, the region pattern of the testing point with respect to the convergin faces of the coincident vertex has an equal number of convex and concave transitions; if between a pair of trivial regions, then the region pattern has two more convex or concave transitions.

Let P_1 be the entrance of the testing vector at R_i , i.e., the intersection of the vector with sphere S at region R_i , and P_2 be the exit at the dual region R^i. We call P_1 the south pole and P_2 the north pole of S , as shown in Figure 29c. We then define a transformation which projects the surface of S from the north pole P_2 onto the tangent plane of the south pole P_1 as shown in Figure 29d. Assume that S is centered at (0,0,0). The transformation is defined as:

$$x = \frac{2rx'}{r - z'}$$
$$y = \frac{2ry'}{r - z'} \tag{4.7}$$

where (x', y', z') is the old coordinate system and (x, y) is the new coordinate system on the tangent plane. The tangent plane, called a *projection plane,* holds the projected

image of the sphere. This mapping is a one-to-one mapping except that the north pole is mapped to infinity. The south pole P_1 is mapped to the origin of the projection plane. Since the trace of v_o is a simple closed curve, the image of the trace on the projection plane is also a closed simple curve which will be called the *projected trace.* The trace of v_o divides the surface of the sphere into two areas; correspondingly, the projected trace divides the projection plane into two regions.

The intersection of a plane in $(G_1, G_2, .., G_m)$ with the sphere S forms a great circle. Therefore, a section of the trace of v_o is a portion of a great circle, i.e., an arc. The projected trace is composed of as many sections as the trace, and each section is an image of an arc. Next we shown that the image of a great circle in this case is still a circle. Therefore, the projected trace is still composed of sections of arcs.

A great circle, formed by the intersection of a sphere with a plane through the center of the sphere, can be described by the equation set

$$\begin{cases} x'^2 + y'^2 + z'^2 = r^2 \\ ax' + by' + z' = 0 \end{cases} \tag{4.8}$$

where $n = (a, b, 1)$ is the normal vector of the plane. Through the transformation in (4.7), the above great circle is projected into a circle

$$(x - 2a)^2 + (y - 2b)^2 = (4 + 4a^2 + 4b^2) \tag{4.9}$$

on the projection plane with radius $= \sqrt{4 + 4a^2 + 4b^2}$ and centered at $= (2a, 2b)$. It is called the *projected circle.* Notice that if a plane is normal to the projection plane, a great circle in that case is mapped to a line instead of a circle. However, such a case does not exist here because we have assumed that a testing vector is not coplanar with any of the planes.

The transformation in (4.7) is a conformal mapping which preserves the angle between two great circles. Let C_1 and C_2 represent two great circles on planes G_1 and G_2 with normal vectors $N_1 =< a_1, b_1, 1 >$ and $N_2 =< a_2, b_2, 1 >$, respectively. They can be described by the set of equations

$$C_1 : \begin{cases} x'^2 + y'^2 + z'^2 = r^2 \\ a_1 x' + b_1 y' + z' = 0 \end{cases} \tag{4.10}$$

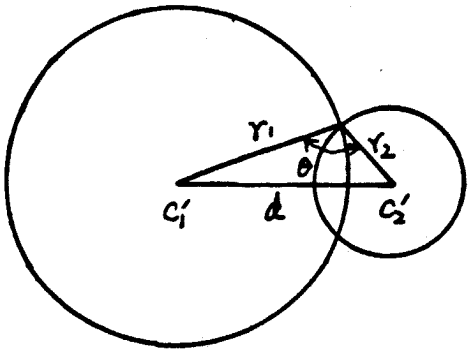

Figure 30

The angle θ between circles C'_1 and C'_2 is the angle between vectors r_1 and r_2 which can be calculated from the radii of C'_1 and C'_2 and the distance between the centers of C'_1 and C'_2.

$$C_2 : \begin{cases} x'^2 + y'^2 + z'^2 = r^2 \\ a_2x' + b_2y' + z' = 0 \end{cases} \tag{4.11}$$

After the transformation, C_1 and C_2 are mapped respectively to C_1' and C_2' on the projection plane:

$$C_1' : (x - 2a_1)^2 + (y - 2b_1)^2 = (4 + 4a_1^2 + 4b_1^2) \tag{4.12}$$

$$C_2' : (x - 2a_2)^2 + (y - 2b_2)^2 = (4 + 4a_2^2 + 4b_2^2) \tag{4.13}$$

The angle Θ between circles C_1 and C_2 is the angle between planes G_1 and G_2. By calculating the inner product of the normal vectors N_1 and N_2, we can find Θ:

$$\cos\Theta = \frac{N_1.N_2}{|N_1|.|N_2|} = \frac{1 + a_1a_2 + b_1b_2}{\sqrt{1 + a_1^2 + b_1^2}\sqrt{1 + a_2^2 + b_2^2}} \tag{4.14}$$

The angle θ between projected circles C'_1 and C'_2, shown in Figure 30, can be calculated as follows.

$$
\begin{aligned}
d &= \text{ the distance between the centers of } C'_1 \text{ and } C'_2. \\
d^2 &= 4(a_1 - a_2)^2 + 4(b_1 - b_2)^2 \\
d^2 &= r_1^2 + r_2^2 - 2r_1 r_2 \cos\theta \\
4(a_1 - a_2)^2 + 4(b_1 - b_2)^2 &= 4(1 + a_1^2 + b_1^2) + 4(1 + a_2^2 + b_2^2) \\
&\quad - 8\sqrt{1 + a_1^2 + b_1^2}\sqrt{1 + a_2^2 + b_2^2}\cos\theta \\
\cos\theta &= \frac{1 + a_1 a_2 + b_1 b_2}{\sqrt{1 + a_1^2 + b_1^2}\sqrt{1 + a_2^2 + b_2^2}}
\end{aligned}
\tag{4.15}
$$

We have thus shown that the transformation indeed preserves the angle between two great circles, i.e., $\Theta = \theta$.

We know that the south pole is mapped to the origin of the projection plane, also that a projected circle always encloses the origin of the projection plane. If the south pole is in the positive half-space of a plane, then the positive half-space is mapped to the interior of the projected circle. For instance in Figure 31a1, the the south pole is in the positive half-space of face F_3, thus the positive half-space is mapped to the interior of the projected circle C_3 as shown in Figure 31a2. Instead, if the south pole is in the negative half-space, then the positive half-space is mapped to the exterior of the projected circle, for example the projected circle C_2 in Figure 31a2. The cross-mapping between the positive/negative half-spaces and the interior/exterior of the projected circle is, therefore, determined by where the south pole is located, and accordingly by where the testing point is located. For example, the testing point in Figure 31a1 is located in the positive half-spaces of both F_1 and F_3, and the negative half-spaces of F_2. The positive half-spaces divided by F_1 and F_3 are thus mapped to the interior of the projected circles C_1 and C_3, respectively. The positive half-space divided by F_2 is then mapped to the exterior of its projected circle, C_2, instead.

If a positive half-space is mapped to the interior of a projected circle, the region pattern bit of the origin with respect to the corresponding arc is positive, as in Figure 32a. If, instead, a negative half-space is mapped to the interior of a projected circle, then the region pattern bit of the origin with respect to the arc is negative, as in Figure 32b. Based on this, the two-dimensional region pattern of the origin of the projection plane with respect to the projected trace can be calculated by calculating the three-dimensional region pattern of the south pole P_1 with respect to the converging

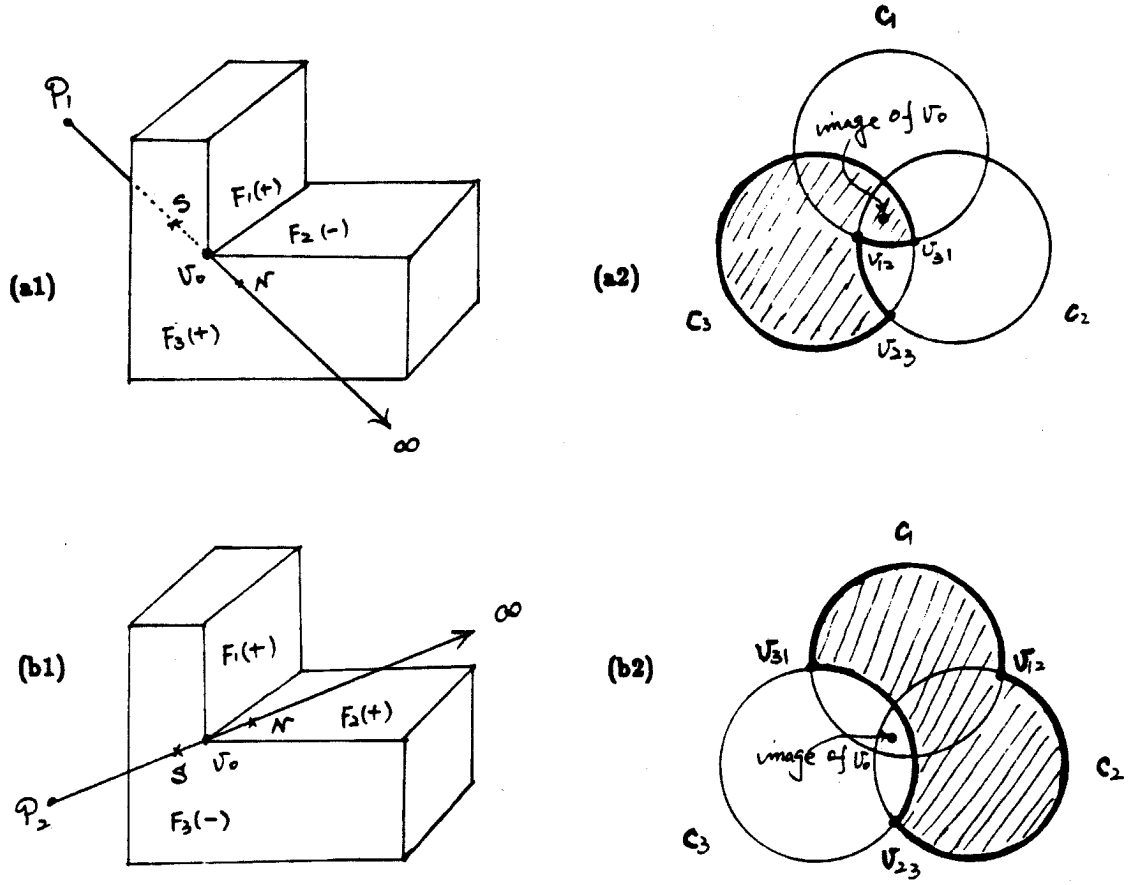

Figure 31

(a1) P_1 is in the $(+,-,+)$ region, and (b1) P_2 is in the $(+,+,-)$ region of faces F_1, F_2 and F_3. The shaded areas correspond, respectively, to (a2) the intersection of the interior of C_3 with the union of the interior of C_1 with the exterior of C_2, and (b2) the intersection of the exterior of C_3 with the union of the interior of C_1 with the interior of C_2.

faces $(F_1, F_2, .., F_m)$ of the coincident vertex. We also notice that the region pattern of the south pole P_1 with respect to $(F_1, F_2, .., F_m)$ is the same as the region pattern of the testing point P with respect to $(F_1, F_2, .., F_m)$. This shows that the region pattern

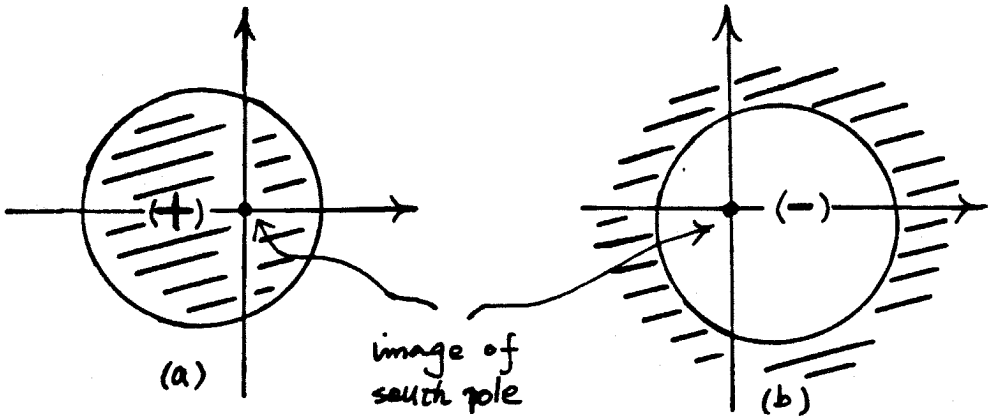

Figure 32
(a) The positive half-space is mapped to the interior of the projected circles if the south pole is located in the positive half-space of a face.
(b) The negative half-space is mapped to the interior of the projected circles if the south pole is located in the negative half-space of a face.

of the origin can be derived from the 3D region pattern of the testing point P with respect to $(F_1, F_2, .., F_m)$.

A convex edge originates from the 'intersection' of two positive half-spaces divided by two adjacent faces, therefore, the image of the interior would correspond to the intersection of the images of the two positive half-spaces. A concave edge derives from the 'union' of two positive half-spaces. The image thus corresponds to the union of the images of two positive half-spaces. For example, F_1 and F_3 in Figure 31a1 intersect a convex edge. the image of the interior of the polyhedron corresponds to the intersection of two projected circles C_1 and C_3, as shown in Figure 33a1. Faces F_1 and F_2 in Figure 31b1 intersect a concave edge. The image thus corresponds to the union of C_1 and C_2, Figure 33b1. However, it is important to point out here that faces F_1 and F_3 in Figure 31b1 also intersect a convex edge. Because the testing point resides in the $(+,-)$ region of F_1 and F_3, the image thus corresponds to the intersection of C_1

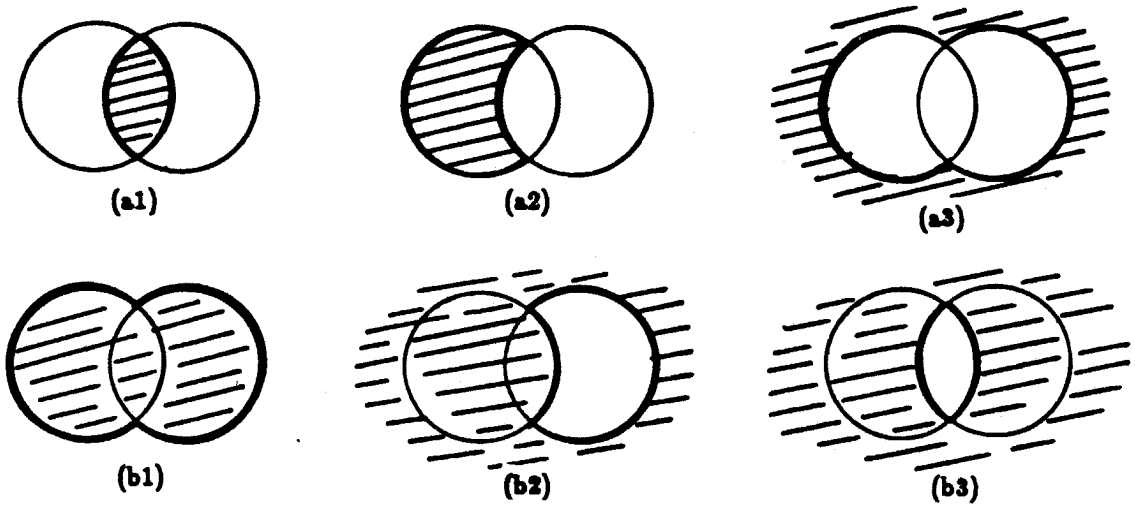

Figure 33

Two faces intersect a convex edge. The south pole is in the (a1) $(+,+)$ region, (a2) $(+,-)$ region, (a3) $(-,-)$ region of the faces. Two faces intersect a concave edge. The south pole is in the (b1) $(+,+)$ region, (b2) $(+,-)$ region, (b3) $(-,-)$ region of the faces.

with the exterior[1] of C_3, as shown in Figure 33a2. F_1 and F_2 in Figure 31a1 intersect a concave edge. However, due to the similar reason, the image corresponds to the union of C_1 with the exterior[2] of C_2, Figure 33b2.

Through the angle-preserving transformation, it is guaranteed that if two consecutive faces in $(F_1, F_2, .., F_m)$ intersect a convex edge, the corresponding arcs in the projected trace also intersect a convex vertex; if two faces intersect a concave edge, the corresponding arcs also intersect a concave vertex. For example, v_{23} in Figure 31a2 is convex because F_2 intersects F_3 by a convex edges, as shown in Figure 31a1. Vertex v_{12} is concave because F_1 intersects F_2 by a concave edge. Therefore, the angle-preserving transformation guarantees that the vertex pattern of the projected trace can be determined from the edge pattern of $(F_1, F_2, .., F_m)$.

---

[1] As we mentioned before, the testing point is in the negative half-space of F_3, the positive half-space of F_3 is thus mapped to the exterior of C_3.

[2] The testing point is in the negative half-space of F_2, the positive half-space of F_2 mapped to the exterior of C_3.

The transition pattern of the origin with respect to the projected trace can be calculated from the vertex pattern of the projected trace and the region pattern of the origin with respect to the projected trace. Since the vertex pattern of the projected trace is equivalent to the edge pattern of $(F_1, F_2, .., F_m)$ and the 2D region pattern of the origin with respect to the projected trace is equivalent to the 3D region pattern of the testing point P with respect to $(F_1, F_2, .., F_m)$, then the 2D transition pattern can be calculated directly from the 3D edge pattern and region pattern.

If both north pole and south pole reside in the same region of the polyhedron (both in the interior or both in the exterior), then they will be mapped to the same region divided by the projected trace. If they reside in opposite regions (one in the interior and the other in the exterior), then they will be mapped to opposite regions divided by the projected trace. We know that the south pole is mapped to the origin and the north pole is mapped to the infinity of the projected plane. Therefore, if north and south poles reside in the same region, then the origin is located ouside of the projected trace, that is, the origin is 'NOT enclosed by' the projected trace. If the poles reside in opposite regions, then the origin will be 'enclosed by' the projected trace.

We mentioned before that a testing vector can travel either from (1) the interior to the exterior, (2) the exterior to the interior, (3) the interior to the interior, or from (4) the exterior to the exterior. If a vector travels from the interior to the exterior or from the exterior to the interior of a polyhedron, it causes a significant intersection with the polyhedron. In this case, as we have shown above, the origin will be 'enclosed by' the projected trace and, therefore, the testing point satisfies the singularity criterion I and has an equal number of convex and concave transitions. If a vector travels from the interior to the interior or from the exterior to the exterior of a polyhedron, it is tangent to the surface of the polyhedron. Therefore, in this case the origin will be excluded by the projected trace and the testing point will fulfill the singularity criterion II and have two more convex transitions than concave transitions or vice versa.

4.7 Discussion

The presence of singularities can confound our attempts at efficient point-polyhedron enclosure detection. This paper shows, however, that the resolution of simple singularities and the decomposition of complex singularities offers not only mathematical elegance but practical utility as well. The singularity criterion help simplify the time-consuming 3D edge-face-penetration detection problem. This makes it particularly useful in practical applications in CAD/CAM.

The development of digital computers has made it possible to carry out computations involving a very large number of arithmetic operations. However, due to the fixed precision of computers, rounding errors could result through iterative addtions and subtractions, [32]. However, according to the discussion in this chapter, we need to keep in mind that the accuracy of the sign of a line equation value is very critical when using the singularity criterion to determine whether a vertor penetrates a surface.

We now consider the rounding errors made in the fundamental arithmetic operations. Addition and subtraction involve no round-off in fixed-point arithmetic, though the possibility of the sum or difference lying outside the permitted range has to be borne in mind. The exact product of two t-digit numbers a and b is, in general, a number requiring $2t$ digits for its representation. This exact product is replaced by the t-digit approximation obtained by adding $\frac{1}{2}2^{-t}$ and discarding the last t digits. Therefore, we have the computational equation

$$c \equiv ab + \epsilon \text{ where } |\epsilon| \leq \frac{1}{2}2^{-t}$$

Nearly most of the digital computers produce, in the first instance, the exact $2t$-digit product of two t-digit numbers. On many computers advantage is taken of this fact to obtain more accurate results. A very common requirement is seen in the calculation of an inner product defined by

$$s = \sum_{1}^{n} a_i b_i$$

If each product in this expression is rounded separately, then the corresponding computational equation is

$$s \equiv \sum_{1}^{n} a_i b_i + \epsilon \text{ where } |\epsilon| \leq \frac{1}{2} n 2^{-t}$$

However, on most of the computers, the inner-product can be accumulated exactly to the full $2t$ figures and the bound for the rounding error is reduced by the factor n:

$$s \equiv \sum_{1}^{n} a_i b_i + \epsilon \text{ where } |\epsilon| \leq \frac{1}{2} 2^{-t}$$

Computers which are provided with this feature have an accumulator which can store a $2t$-digit number. However, the details of the fundamental floating-point arithmetic operations differ from one computer to another. On a computer having a double-precision accumulator, the operations are done as follows. Let the two floating numbers x_1 and x_2 be denoted by $x_1 = 2^{b_1} a_1$ and $x_2 = 2^{b_2} a_2$. We also assume that x_1 is the number with the larger modulus. Then the integer $(b_1 - b_2)$ is computed: (i) if $(b_1 - b_2) > t$, the number of digits, then x_2 is too small to have any effect as far as the first t significant digits of the sum are concerned; (ii) if $(b_1 - b_2) \leq t$, then a_2 is divided by $2^{b_1 - b_2}$ by shifting it $b_1 - b_2$ places to the right. The sum $a_1 + 2^{b_1 - b_2} a_2$ is then calculated exactly. This sum is then multiplied by the appropriate power of 2 so that the resulting number lies in the range permitted for the mantissa of a floating-point number.

Therefore, if the difference between b_1 and b_2 is more than the number of digits t, then one can loose the precesion on the result as much as x_2. In calculating the sign of a line equation value $ax + by + cz + d$, this effect can sometime lead to an erroneous result.

Chapter 5

Polyhedron-Polyhedron Intersection

The set operation has a central role in computer graphics problems where the goal is to construct a new polygon (or polyhedra) out of two separate originals. We can define the subspace occupied by the resultant polygon (or polyhedra) as the union, intersection, or difference of the subspaces occupied by the two original polygons (or polyhedra); and conventionally we can execute the necessary computation by means of a dot matrix scheme where each dot represents the occupancy of a particular small space by the object. With this method, performing a set operation on two objects becomes a very simple task: we merely select the appropriate set of dots out of the two matrices of the objects. For instance, to perform an *intersection* operation on two objects, we select the dots which are present in both objects; to perform a *union* operation on two objects, we select the dots which are present in either object. Simple though this method is theoretically, however, practically it demands high memory costs.

A more efficient way of representing a solid is by describing its boundary; this is known as the boundary representation scheme. Sufficient information about the boundary can uniquely identify the space occupied by the solid. Performing a set operation on two objects represented with boundary representation scheme then becomes a task of constructing the new boundary of the resultant object according to the specifications demanded by the specific operation classes.

With boundary representation scheme, however, polygon-polygon intersection is the first step to perform in a set operation between two polygons. Polygon-polygon intersection detection is an operation on two polygons which compares every edge of one polygon against every edge of the other polygon to find the intersections between them, as shown in Figure 34a. This edge comparison between two polygons is called edge-edge intersection detection and is for determining whether two edges intersect each other, as in Figure 34b. If they do intersect, the operation further discovers

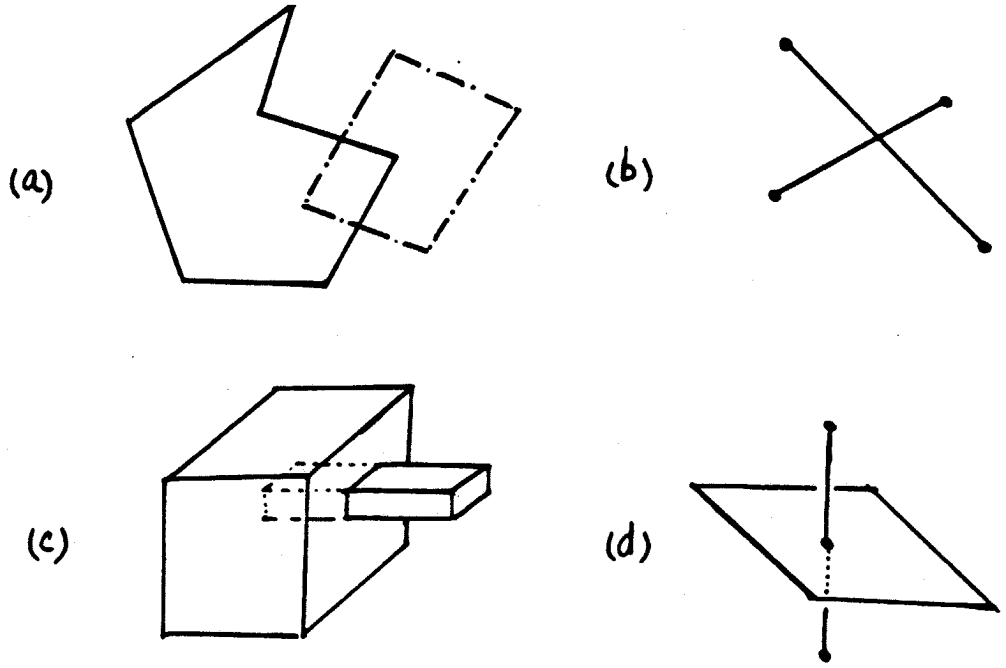

Figure 34
(a) Polygon-polygon intersection, (b) edge-edge intersection, (c) polyhedron-polyhedron intersection, (d) edge-face penetration.

the coordinates of the intersection point. A polygon-polygon intersection thus consists of n^2 independent edge-edge intersections, where n is the number of edges in one polygon. Sometimes, the number of edge-edge intersection detections can be reduced by pre-checking the maximum and minimum coordinates of the vertices of the two edges.

Similarly, polyhedron-polyhedron intersection is the first step in performing a set operation on two polyhedra in a three-dimensional space. A polyhedron-polyhedron intersection is an operation which compares every face of one polyhedron against every face of the other polyhedron, to find all the intersections between the faces of the two polyhedra, as in Figure 34c. This face comparison is composed of multiple independent edge-face-penetration detections, each of which compares an edge of a face with another face to determines whether the edge penetrates the face, as in Figure 34d. An edge-face-penetration problem can be solved by a 2D point-polygon-enclosure detec-

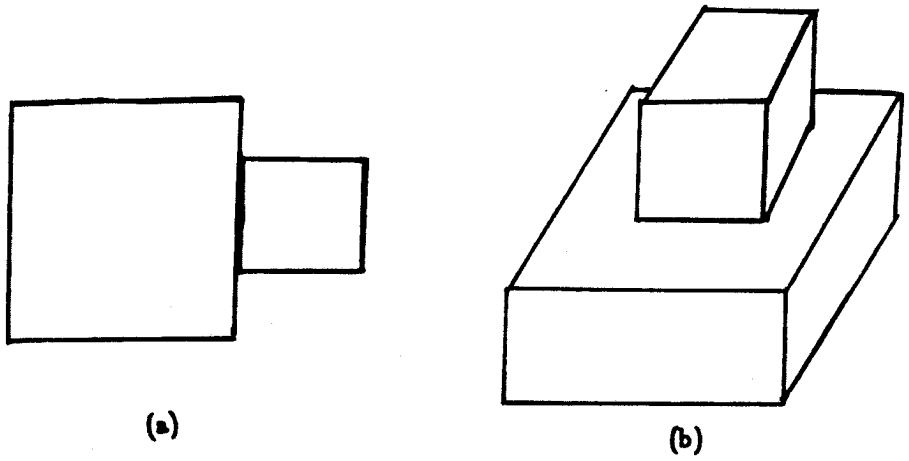

Figure 35

(a) Two polygons intersect by a line segment, (b) two polyhedra intersect by a face.

tion method or using the singularity criterion proposed in Chapter 4. A polyhedron-polyhedron intersection consists of $2mn$ independent edge-face penetrations, where n is the number of edges and m is the number of faces of a polyhedron.

As one can see, polygon-polygon intersection and polyhedron-polyhedron intersection are the two most computation-intensive operations in performing a set operation on two objects. In the following discussion we describe how to perform these operations in an easier and more efficient way. A perturbation method is suggested below which can be used to transform two polygons (or two polyhedra) into a singularity-free situation so that polygon-polygon intersection and polyhedron-polyhedron intersection can be performed without the necessity of considering degenerate situations.

5.1 Singularities

When performing a set operation on two objects some undesirable situations may occur: for instance, two polygons may intersect by a line segment, as in Figure 35a; or two polyhedra may intersect by a face, as in Figure 35b. Situations like these are called "degenerate situations" or "singularities." Singularities are a very troublesome but inevitable problem. They are troublesome because their result is degenerate, for

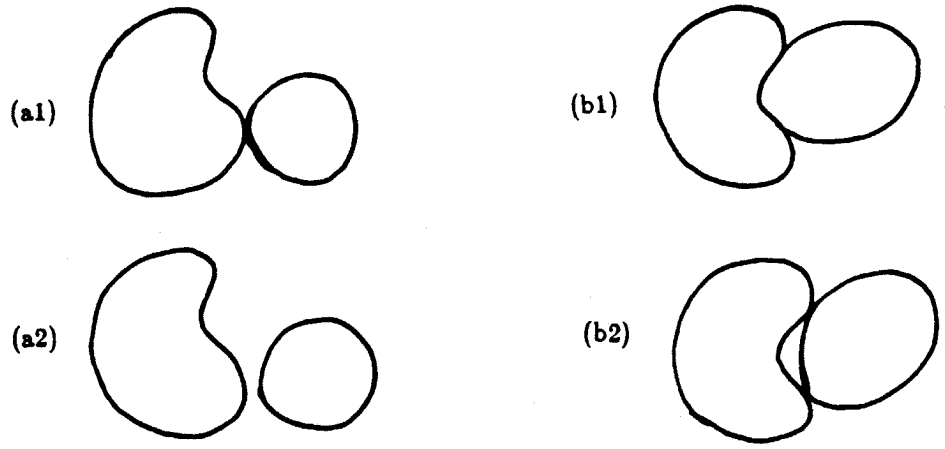

Figure 36

(a1) Two polygons are tangent to each other, (a2) two polygons are detached from each other, (b1) part of the boundaries of the two polygons overlap with each other, (b2) the overlapping edge portions are detached from each other.

instance, the intersection of the two polygons is a line segment which does not have an area; and the intersection of the two polyhedra is a face which does not have a volume. Such degenerate situations usually cause erroneous results if they are not carefully handled. A conventional way to handle them is to allow the presence of partial intersections, such as a half or a quarter intersection, depending on the nature of the singularity.

Let us first consider a generally defined polygon which is a smooth closed curve instead of a segmented polygon. When we intersect two polygons like this, we may face the following difficult situations: (1) the boundaries of the two polygons are tangent to each other, as in Figure 36a1, or (2) part of the boundaries of the two polygons overlap with each other, as in Figure 36b1. The former case is a zero-order singularity which will be called a 'D^0 singularity', and the latter one is a first-order singularity, which will be called a 'D^1 singularity.' In two different cases of these two singularities, the overlapping boundaries of P and Q face either the same or opposite directions.

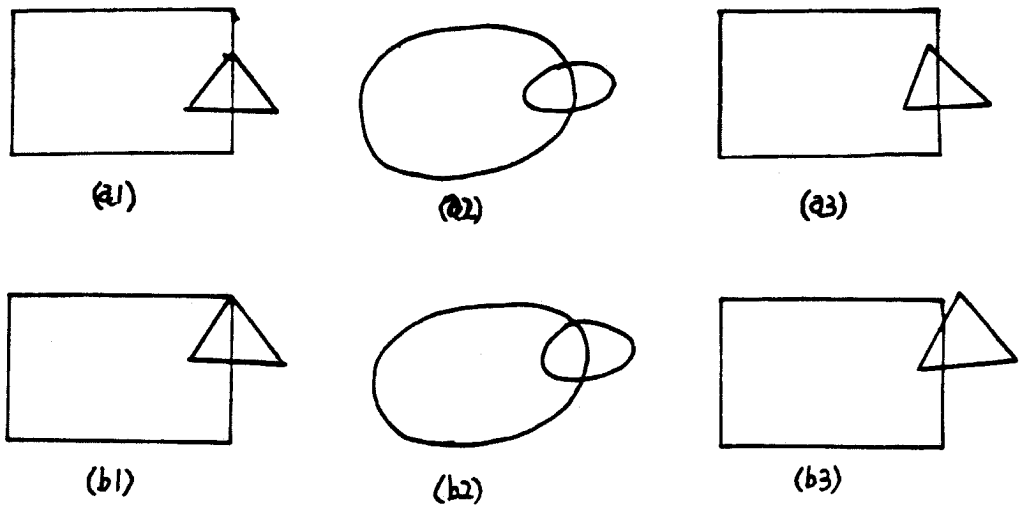

Figure 37

(a1) A virtual V^o singularity, (b1) a virtual V^{oo} singularity, (a2) smoothed polygon, (b2) smoothed polygon, (a3) the coincident vertex turned slightly away from the edge, (b3) the coincident vertex turned slightly away from the corner.

As we can see, we can easily avoid a zero-order singularity by perturbing the coincident vertex just enough to detach it from the coincident edge, as in Figure 36a2. Similarly, a first-order singularity can be transformed into two lower order singularities by slightly perturbing and thus detaching the overlapping edges, as in Figure 36b2. In this case, the singularity can then by resolved by digesting its two components.

When considering segmented polygons, other degenerate situations may occur in addition to the degenerate cases mentioned above. These degenerate situations are considered normal when polygons are smoothed. For instance, the situations in Figure 37a1 and 37b1 are considered degenerate only because the vertex is located on a polygon. However, as we can see in Figure 37a2 and 37b2, they are not real degenerate situations once the polygons are smoothed. These cases are *virtual* zero-order singularities. The one in (a1) is called a virtual V^o singularity, and the one in (b1) is a virtual V^{oo} singularity. We can tell a virtual singularity from the fact that each polygon subdivides the interior of the other polygon. These degeneracies can

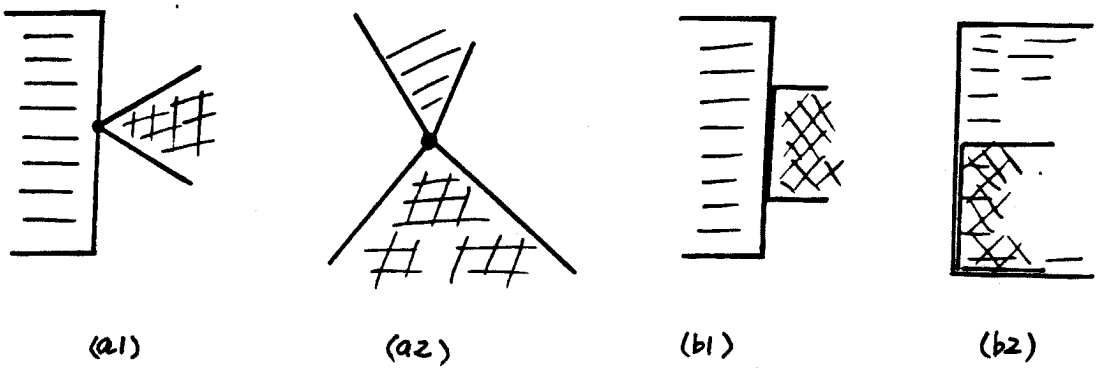

Figure 38

(a1) A D^0 singularity, (a2) a D^{00} singularity, (b1) a D^1 singularity, (b2) a D^{11} singularity.

usually be avoided by virtually moving one coincident vertex slightly away from the other so that they turn into normal situations, as in Figure 37a3 and 37b3. The term "virtually moving the vertex" implies that no physical movement is really enforced onto the vertex; we only assume the vertex has been moved.

Because of the segmentation at the boundaries of the polygons, a D^0 singularity can turn into a D^{00} singularity, as shown in Figure 38a, when the vertices of two polygons overlap; and a D^1 singularity can turn into a D^{11} singularity, as shown in Figure 38b, when two edges of the two polygons overlap. A D^0 singularity and a D^{00} singularity are thus basically the same when considered in a smooth boundary polygon, and so are a D^1 and a D^{11} singularities.

Next we discuss how to handle singularities encountered in performing a set operation on two polygons. Here the polygons are segmented polygons whose boundaries are composed of a sequence of line segments called edges. In a set operation on two polygons P and Q, we first perform an edge-edge-cross examination between the two polygons. The intersection of two segments can be determined by calculating the line-equation values of both ends of one segment against the other segment, and vice versa. We use a function "/" (called "*against*") to determine whether a point P is located on the positive side or the negative side of an edge, or whether the point falls

right on the line which contains the edge. The function maps a point $P =< x_o, y_o >$ and a line $L : ax + by + c = 0$ into an integer in $\{+1, 0, -1\}$. It returns a value "1" if a point is located on the positive side, a value "-'1' if it is on the negative side. If P falls on the line, then the function returns a value "0". Therefore, we can define the function as the sign of the line-equation value as $P/L = 1$ if $ax_o + by_o + c > 0$; $P/L = -1$ if $ax_o + by_o + c < 0$; $P/L = 0$ if $ax_o + by_o + c = 0$.

$$P/L \quad = \quad \text{Sign}(ax_o + by_o + c) \tag{5.1}$$

Let edges $E_p = (v_1, v_2)$ and $E_q = (v_3, v_4)$ be edges of P and Q, respectively, in comparison. Let L_p and L_q represent the two lines containing edges E_p and E_q, respectively. To determine whether E_p intersects E_q, v_1 and v_2 are compared "against" E_q and v_3 and v_4 are compared "against" E_p. We know that vertices v_1 and v_2 are on the opposite sides of edge E_q when $(v_1/L_q)(v_2/L_q) < 0$, and v_3 and v_4 are on the opposite sides of edge E_p when $(v_3/L_p)(v_4/L_p) < 0$. Edges E_p and E_q intersect when v_1 and v_2 are on the opposite sides of edge E_q and v_3 and v_4 are on the opposite sides of edge E_p, such that

$$(v_3/L_p)(v_4/L_p) < 0 \text{ and } (v_1/L_q)(v_2/L_q) < 0. \tag{5.2}$$

In this case we say that the two edges intersect "normally."

However, in a singular situation, two edges intersect when one end of an edge is on the other edge:

$$(v_3/L_p)(v_4/L_p) \leq 0 \text{ or } (v_1/L_q)(v_2/L_q) \leq 0$$

We call this case a singular situation, or a singularity. A singularity occurs when v_1 or v_2 is located on E_q or v_3 or v_4 is located on E_p, or sometimes when E_p and E_q are overlapping with each other:

$$\begin{aligned}
&(v_3/L_p)(v_4/L_p) < 0 \text{ and } (v_1/L_q)(v_2/L_q) = 0, \\
&(v_3/L_p)(v_4/L_p) = 0 \text{ and } (v_1/L_q)(v_2/L_q) < 0, \text{ or} \\
&(v_3/L_p)(v_4/L_p) = 0 \text{ and } (v_1/L_q)(v_2/L_q) = 0
\end{aligned}$$

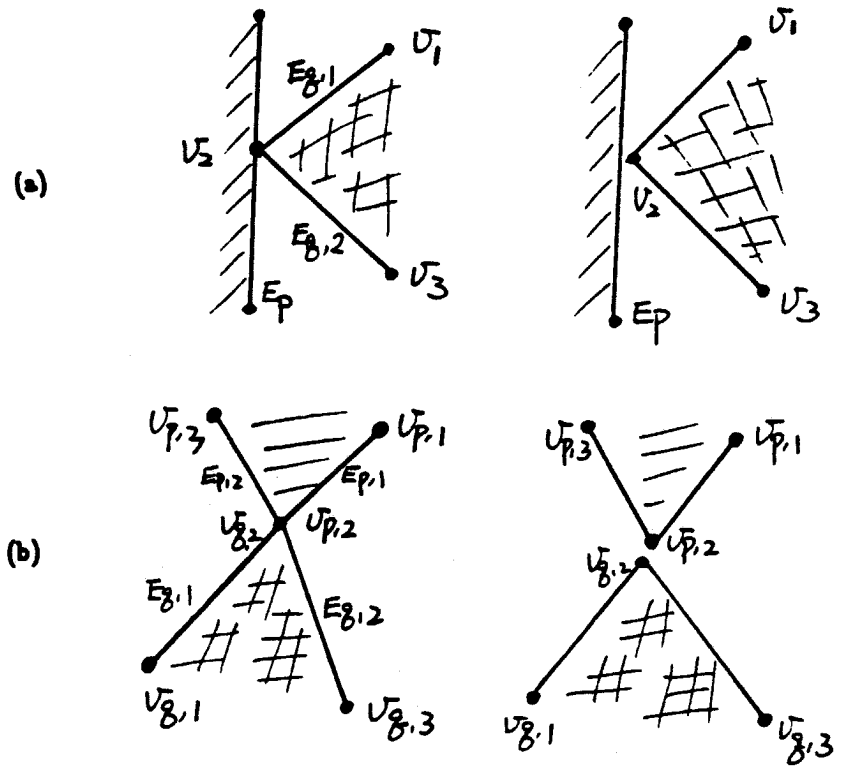

Figure 39

(a) Vertex v_2 is moved away from E_p by a perturbation so that the D^o singularity disappears. (b) Vertices $v_{p,2}$ and $v_{q,2}$ are detached from each other by a perturbation so that the D^{oo} singularity disappears.

Here we say that the two edges intersect in a singular situation. To remove it, we would like to enforce a perturbation on one of the polygons during the edge-edge-cross examination.

Vertex v_2 in Figure 39a is a vertex of polygon Q which causes a D^o singularity on an edge E_p of polygon P with condition $v_2/E_p = 0$. A slight perturbation can cause v_2 to move away from E_p into the same region where v_1 is located. Vertices v_1, v_2, and v_3 are thus located in the same region divided by the line containing E_p, and clearly there is no intersection between $E_{q,1}$ with E_p, nor between $E_{q,2}$ with E_p. Therefore, the perturbation causes the singularity $(v_1/E_p)(v_2/E_p) = 0$ and $(v_3/E_p)(v_2/E_p) = 0$ to turn into $(v_1/E_p)(v_2/E_p) > 0$ and $(v_3/E_p)(v_2/E_p) > 0$.

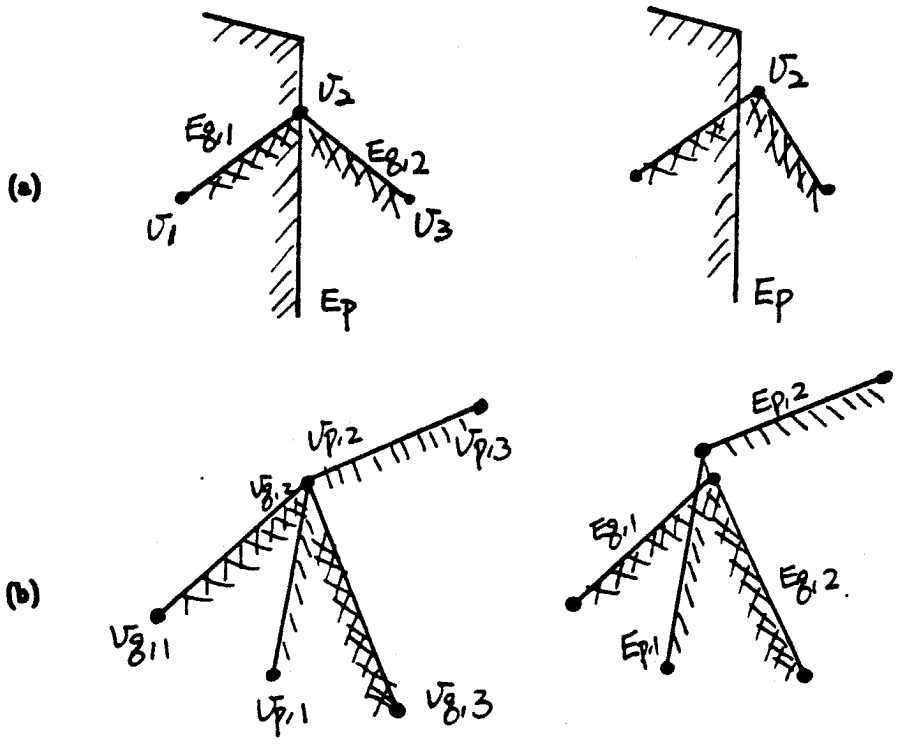

Figure 40

(a) Vertex v_2 is moved into the positive region of E_p so that the V^o singularity is no longer present, (b) vertices $v_{p,2}$ and $v_{q,2}$ are shifted away from each other to avoid the V^{oo} singularity.

In Figure 39b vertices $v_{p,2}$ of polygon P and $v_{q,2}$ of polygon Q coincide and result in a D^{oo} singularity with conditions $v_{p,2}/E_{q,1} = 0$, $v_{p,2}/E_{q,2} = 0$, $v_{q,2}/E_{p,1} = 0$, $v_{q,2}/E_{p,2} = 0$. Since the interiors of the two polygons are disjoint, a perturbation can detach $v_{q,2}$ from $v_{p,2}$ so that the two corners are isolated from each other and the singularity simply disappears. Thereafter, neither $E_{p,1}$ or $E_{p,2}$ intersects $E_{q,1}$ or $E_{q,2}$.

Vertex v_2 in Figure 40a causes a V^o singularity because v_2 is located on edge E_p and v_1 and v_3 are on the opposite sides of E_p. The perturbation moves v_2 into the positive region of E_p, thus avoiding the singularity. In other words, the perturbation enforces $v_2/E_p = 0$ into $v_2/E_p = +1$ and resulting in the situation that

$$(v_1/E_p)(v_2/E_p) < 0 \text{ and } (v_2/E_p)(v_3/E_p) > 0$$

We then know that there is a normal intersection between $E_{q,1}$ with E_p, but no intersection between $E_{q,2}$ with E_p. However, it is important to mention here that the coordinates of the intersection point between $E_{q,1}$ and E_p should be the same as the coordinates of point v_2, so that we assure there is no physical displacement really enforced on the polygon. The perturbation is only a virtual means to determine which two edges do intersect and which two do not.

By this same reasoning we can resolve the V^{oo} singularity in Figure 40b with a perturbation, shifting $v_{p,2}$ and $v_{q,2}$ away from the edge and moving the intersection onto edges $E_{p,1}$ and $E_{q,2}$. The perturbation leaves the result that only edges $E_{p,1}$ and $E_{q,2}$ have normal contact, but not edges $E_{p,1}$ with $E_{q,1}$, $E_{p,2}$ with $E_{q,1}$, or $E_{p,2}$ with $E_{q,2}$. Again, the coordinates of the intersection point of $E_{p,1}$ with $E_{q,2}$ should be the same as the coordinates of points $v_{p,2}$. In some cases there may be no physical perturbation to displace intersections onto edges $E_{p,1}$ and $E_{q,1}$ while leaving no intersection between the rest of the edges. However, since no physical displacement is really enforced onto the polygons, assuming the existence of such a nonrealizable perturbation would still lead to a satisfactory result.

A D^1 singularity can be transformed into two lower order singularities, either two D^o singularities, a D^o and a V^o singularity, or two V^o singularities, by pushing the coincident edge of the second polygon, E_q into the positive side of the first polygon, and then resolving its two components following the previous rules. A D^{11} singularity can be resolved in a similar way by pushing the coincident edge of the second polygon into the interior of the first polygon.

The method discussed above for avoiding singularities is called the "neighbors look-up" method, because the perturbation enforced depends on the type of neighbors a coincident vertex has. For instance, a vertex v_2 of polygon P is located on an edge E_q of polygon Q. Vertex v_2 may be in the middle of several kinds of singularities such as a D^o singularity, a V^o singularity, a D^{oo} singularity, a V^{oo} singularity, or even a D^1 singularity, see Figure 41e. We would perturb vertex v_2 differently depending on what kind of singularity it is in.

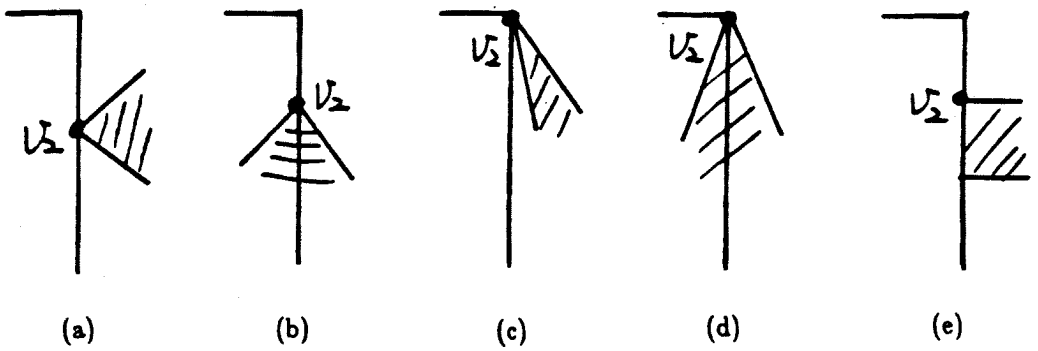

Figure 41
(a) A D^0 singularity, (b) a V^0 singularity, (c) a D^{00} singularity, (d) a V^{00} singularity, (e) a D^1 singularity.

5.2 Edge-edge intersection

Because the number of degenerate cases increases enormously when we enter a 3D space, we can expend a great effort trying to admit every possible singularity and solving each of them on a case-by-case basis while performing the set operation on two objects. Next we discuss a singularity-independent perturbation method which can "virtually" eliminate all the singularities encountered in polygon-polygon intersection or polyhedron-polyhedron intersection. This method resolves a singularity independent of the neighbors of the singularity-causing element. It resolves singularities at the level of edge-edge intersection (or the edge-face penetration) so that singularities can be handled at this step independent of the type of operation being performed.

In the following discussion we explain the details of this method and show how singularities can be solved by using the idea of "perturbation." As we explained previously, perturbation is a method of slightly displacing one polygon (polyhedron) relative to another. One might think that in order to create an infinitely small perturbation one would need to make an infinitely small change on the coordinates of vertices, which is difficult to achieve on a computer. However, perturbation is simply a way used to develop a set of rules which one can follow whenever a singularity is

encountered. At the end of this section such a set of rules is presented. These rules are the core of the singularity-independent perturbation method. As far as implementation is concerned, these rules are what we need to code a "singularity-resolver."

As we described before, the intersection of two edges can be determined by calculating the line equation values of both ends of one edge against the other edge, and vice versa. If both ends of the first edge are located at opposite sides of the second edge, and both ends of the second edge are also located at opposite sides of the first edge, we know that the two edges intersect.

Let Q_a and Q_b be the two polygons in operation. Let $E_a = (V_a^1, V_a^2)$ and $E_b = (V_b^1, V_b^2)$ be edges of Q_a and Q_b , respectively, under edge-edge intersection examination. Let the coordinates of the vertices be $V_a^1 =< x_a^1, y_a^1 >$, $V_a^2 =< x_a^2, y_a^2 >$, $V_b^1 =< x_b^1, y_b^1 >$, and $V_b^2 =< x_b^2, y_b^2 >$. Let

$$L_a : \alpha_a x + \beta_a y + \gamma_a = 0$$
$$L_b : \alpha_b x + \beta_b y + \gamma_c = 0$$

represent the lines containing edges E_a and E_b , respectively. Two edges intersect normally if the following condition stands:

$$(V_a^1/L_b)(V_a^2/L_b) < 0 \text{ and } (V_b^1/L_a)(V_b^2/L_a) < 0 \tag{5.3}$$

If there are singularities present, we assume that the second object Q_b is displaced by an infinitely small perturbation (ϵ_x, ϵ_y) relative to the first object Q_a so that all the singularities disappear. In other words, we like to choose a perturbation (ϵ_x, ϵ_y) which can perturb the two objects so that they intersect each other in a singularity-free situation. We also assume that both ϵ_x and ϵ_y are positive and that the order of magnitude of ϵ_x is far larger than ϵ_y, so that the ϵ_y perturbation on y-axis can be neglected whenever the ϵ_x perturbation on x-axis is not negligible.

For example, a singularity occurs when vertex V_b^1 is located on line L_a, so that the line-equation value is zero.

$$(V_b^1/L_a) = \alpha_a x_b^1 + \beta_a y_b^1 + \gamma_a = 0$$

With an small perturbation (ϵ_x, ϵ_y) on Q_b , vertex $V_b^1 =< x_b^1, y_b^1 >$ is displaced to a new position $V_b^1 =< x_b^1 + \epsilon_x, y_b^1 + \epsilon_y >$, so that the line-equation value is not anymore zero. Since we are concerned with only the sign of the line-equation value, we can express (V_b^1/L_a) as

$$(V_b^1/L_a) = \text{Sign}(\alpha_a \epsilon_x + \alpha_b \epsilon_y)$$

Since both ϵ_x and ϵ_y are positive and the effect of ϵ_y is negligible when compared to ϵ_x, we conclude that

$$\begin{aligned} (V_b^1/L_a) &= \text{Sign}(\alpha_a) \text{ if } \alpha_a \neq 0 \\ (V_b^1/L_a) &= \text{Sign}(\beta_a) \text{ if } \alpha_a = 0 \end{aligned} \tag{5.4}$$

If the x-component of the normal vector $\alpha_a \neq 0$, the line L_a is not a horizontal line. Applying only ϵ_x perturbation on the x-axis alone can shift the coincident vertex V_b^1 away from L_a. However, if L_a is a horizontal line, that is if the normal vector has $\alpha_a = 0$, the coincident vertex V_b^1 is still on line L_a if it is perturbed only on the x-axis, no matter by what amount. In order to shift the coincident vertex V_b^1 away from L_a, the ϵ_y perturbation on y-axis is necessary. However, the magnitude of ϵ_y is restricted not to overtake the effect of ϵ_x, so that the signs of $\alpha_a \epsilon_x + \alpha_b \epsilon_y$ and $\alpha_a \epsilon_x$ can be identical.

A similar rule applies to singularities occurring on vertex $V_b^2 =< x_b^2, y_b^2 >$ when $(V_b^2/L_a) = 0$: the perturbation causes V_b^2 to be displaced to a new position $V_b^2 =< x_b^2 + \epsilon_x, y_b^2 + \epsilon_y >$, so that

$$\begin{aligned} (V_b^2/L_a) &= \text{Sign}(\alpha_a) \text{ if } \alpha_a \neq 0, \text{ otherwise} \\ (V_b^2/L_a) &= \text{Sign}(\beta_a) \end{aligned} \tag{5.5}$$

However, a singularity may occur on the vertices of polygon Q_a instead of Q_b, for instance, on vertex V_a^1 with condition

$$(V_a^1/L_b) = \alpha_b x_a^1 + \beta_b y_a^1 + \gamma_b = 0$$

Since polygon Q_b is perturbed by $< \epsilon_x, \epsilon_y >$ relative to polygon Q_a , we can say that polygon Q_a is perturbed by $< -\epsilon_x, -\epsilon_y >$ relative to polygon Q_b. Therefore,

instead of shifting the line L_b by (ϵ_x, ϵ_y), we can imagine that vertex V_a^1 is displaced by $(-\epsilon_x, -\epsilon_y)$ to $V_a^1 =< x_a^1 - \epsilon_x, y_a^1 - \epsilon_y >$. Therefore, we arrive at the following conclusion when singularities occur on the vertices of polygon Q_a instead of Q_b :

$$(V_a^1/L_b) = \text{Sign}(-\alpha_b) \text{ if } \alpha_b \neq 0, \text{ otherwise}$$
$$(V_a^1/L_b) = \text{Sign}(-\beta_b)$$

For a singularity occurring on vertex $V_a^2$of Q_a , the perturbation also leads to a similar situation

$$(V_a^2/L_b) = \text{Sign}(-\alpha_b) \text{ if } \alpha_b \neq 0, \text{ otherwise}$$
$$(V_a^2/L_b) = \text{Sign}(-\beta_b) \tag{5.6}$$

By following the above rules, edge-edge intersection can be strictly determined by the following condition:

$$(V_a^1/L_b)(V_a^2/L_b) < 0 \text{ and } (V_b^1/L_a)(V_b^2/L_a) < 0$$

5.3 Face-face intersection

An edge-face-penetration detection can be performed by first checking whether both ends of the edge are located on the opposite sides of the plane containing the face, and then checking whether the intersection point of the edge with the plane is enclosed by the face. The second operation is called 2D point-polygon-enclosure detection. If both conditions hold true, then we know that the edge intersects the face. Thus each edge-face-penetration detection ideally requires one 2D point-polygon-enclosure detection. In practical application, however, it would take much computer time to compare every edge of the first polyhedron against every face of the second polyhedron and also every edge of the second polyhedron against every face of the first polyhedron.

Face-face intersection compares a pair of faces to determine whether they intersect. Instead of decomposing a face-face intersection detection into multiple edge-face-penetration detections as the conventional technique does, a simpler method [9] can be used which takes as many plane-equation calculations as the total number of

edges of the two faces in comparison to determine the intersection. Next we discuss how to apply the perturbation method in case of singularities in 3D face-face intersection detection.

Let Q_a and Q_b be the two polyhedra in operation. Let F_a be a face of polyhedron Q_a composed of m edges, $F_a = (E_a^1, E_a^2, .., E_a^m)$ and let F_b be a face of polyhedron Q_b, composed of n edges, $F_b = (E_b^1, E_b^2, .., E_b^m)$. Each edge is described by two vertices, $E_a^i = (v_a^i, v_a^{i+1})$ and $E_b^j = (v_b^j, v_b^{j+1})$. Let G_a and $G_b$be the two planes containing faces F_a and F_b, respectively, such that

$$G_a : \alpha_a x + \beta_a y + \gamma_a z + \delta_a = 0$$
$$G_b : \alpha_b x + \beta_b y + \gamma_b z + \delta_b = 0$$

If planes G_a and G_b are not parallel to each other, the intersection line $L_{a,b}$ between G_a and G_b can be described by an equation set

$$L_{a,b} : \begin{cases} \alpha_a x + \beta_a y + \gamma_a z + \delta_a = 0 \\ \alpha_b x + \beta_b y + \gamma_b z + \delta_b = 0 \end{cases}$$

We use a function "/" ("*against*") which maps a point $P =< x_o, y_o, z_o >$ and a plane (or a face) $G : \alpha x + \beta y + \gamma z + \delta = 0$ into an integer in $\{+1, 0, -1\}$. The function returns "1" if the point P is located in the positive side of the plane; or "-1" if the point is in the negative side; if the point is in the plane then it returns "0." We define the function by $P/G = 1$ if $\alpha x_o + \beta y_o + \gamma z_o + \delta > 0$, by $P/G = -1$ if $\alpha x_o + \beta y_o + \gamma z_o + \delta < 0$, or by $P/G = 0$ if $\alpha x_o + \beta y_o + \gamma z_o + \delta = 0$, such that

$$P/G \quad = \quad \text{Sign}(\alpha x_o + \beta y_o + \gamma z_o + \delta) \tag{5.7}$$

If an edge of F_a intersects the plane G_b, it makes a cut on the line $L_{a,b}$; so does an edge of $F_b$if it intersects the plane G_a . If an edge $E_a = (v_a^1, v_a^2)$ of F_a intersects plane G_b, both ends of the edge must be on the opposite sides of the plane, which can be described by the condition

$$(v_a^1/G_b)(v_a^2/G_b) < 0 \tag{5.8}$$

If an edge $E_b = (v_b^1, v_b^2)$ of F_b intersects plane G_a , it must satisfy the condition

$$(v_b^1/G_a)(v_b^2/G_a) < 0 \tag{5.9}$$

Therefore, an edge of F_a makes a cut on $L_{a,b}$ when condition (5.8) is satisfied; and so does an edge of F_b when condition (5.9) is satisfied.

In the event that edge E_b^j intersects plane G_a in a singular condition where

$$(v_b^j/G_a) = 0 \quad \text{or} \quad (v_b^{j+1}/G_a) = 0$$

we apply an infinitely small perturbation $< \epsilon_x, \epsilon_y, \epsilon_z >$ to the second polyhedron Q_b to displace vertex v_b^j to $v_b^j =< x_b^j + \epsilon_x, y_b^j + \epsilon_y, z_b^j + \epsilon_z >$, so that

$$v_b^j/G_a \quad = \quad \text{Sign}(\alpha_a\epsilon_x + \beta_a\epsilon_y + \gamma_a\epsilon_z)$$

We also assume that ϵ_x, ϵ_y and ϵ_z are positive and the effect of ϵ_x can overcome that of ϵ_y, and the effect of ϵ_y can overcome that of ϵ_z. We conclude that

$$\begin{aligned}
&v_b^j/G_a = \text{Sign}(\alpha_a) \text{ if } \alpha_a \neq 0 \text{ else} \\
&v_b^j/G_a = \text{Sign}(\beta_a) \text{ if } \alpha_a = 0, \beta_a \neq 0 \text{ else} \\
&v_b^j/G_a = \text{Sign}(\gamma_a) \text{ if } \alpha_a = \beta_a = 0
\end{aligned}$$

However, if a singularity occurs on polygon Q_a instead of Q_b , i.e., an edge E_a^i of F_a intersects plane G_b, such that

$$(v_a^i/G_b) = 0 \quad \text{or} \quad (v_a^{i+1}/G_b) = 0$$

then the perturbation leads to the situations

$$\begin{aligned}
&v_a^i/G_b = \text{Sing}(-\alpha_b) \text{ if } \alpha_b \neq 0 \text{ else} \\
&v_a^i/G_b = \text{Sign}(-\beta_b) \text{ if } \alpha_b = 0, \beta_b \neq 0 \text{ else} \\
&v_a^i/G_b = \text{Sign}(-\gamma_b) \text{ if } \alpha_b = \beta_b = 0
\end{aligned}$$

Since a face is a simple closed curve, an even number of cuts is always made on the cut line $L_{a,b}$ by each face, as Figure 42 shows. Let $(\mu_1, \mu^1, ..., \mu_r, \mu^r)$ be the set of $2r$ cuts made by face F_a , and let $(\nu_1, \nu^1, ..., \nu_s, \nu^s)$ be the set of $2s$ cuts made by face F_b, where the cuts are sorted according to their position on the cut line $L_{a,b}$, as in Figure 43a. We know that the cuts are in a paired structure so that each pair (μ_i, μ^i)

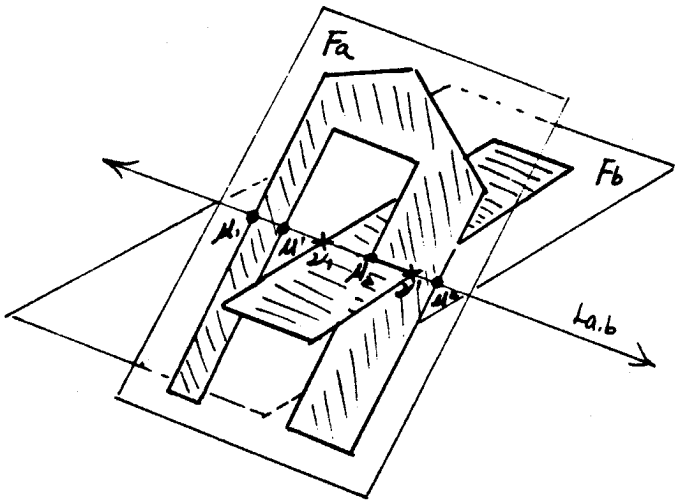

Figure 42

Four cuts $(\mu_1, \mu^1, \mu_2, \mu^2)$ are made on the cut line $L_{a,b}$ by F_a. Two cuts (ν_1, ν^1) are made by face $F_b$on the cut line $L_{a,b}$.

constitutes a segment that occupies the interior of face F_a , as in Figure 43b, and each pair (ν_j, ν^j) constitutes a segment that occupies the interior of face F_b, as in Figure 43c. However, two consecutive cuts from two neighboring pairs such as (μ^i, μ_{i+1}) or (ν^j, ν_{j+1}) constitute a segment occupying the exterior of the face. By sorting the two sequences of cuts $(\mu_1, \mu^1, ..., \mu_r, \mu^r)$ and $(\nu_1, \nu^1, ..., \nu_s, \nu^s)$ together, we can determine which sections of the cut line $L_{a,b}$ form the intersections of F_a with F_b, as in Figure 43d.

We know that if a segment of $L_{a,b}$ forms the intersection of F_a and F_b, then that segment must occupy the interiors of both F_a and F_b. Therefore, we know that if a segment $S_a^i = (\mu_i, \mu^i)$ is in face F_b, then S_a^i forms part of the intersection of F_a with F_b. Conversely, if a segment $S_b^j = (\nu_j, \nu^j)$ is in face F_a , then S_b^j also forms part of the intersection of F_a with F_b.

When the cut line is considered as a one-dimensional space, we can compare two points p_1 and p_2 on the cut line by their positions: say $p_1 < p_2$ if p_1 is on

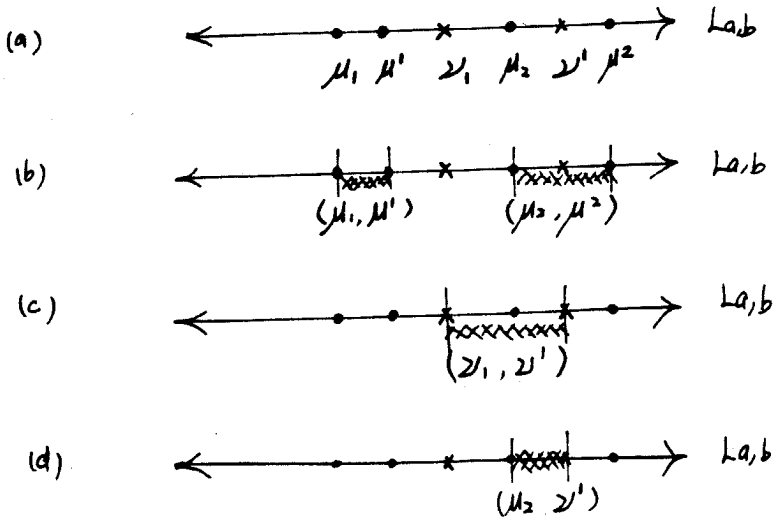

Figure 43

(a) Four cuts $(\mu_1, \mu^1, \mu_2, \mu^2)$ made by F_a and two cuts (ν_1, ν^1) made by $F_b$are sorted according to their position on the cut line $L_{a,b}$, (b) Segments (μ_1, μ^1) and (μ_2, μ^2) occupy the interior of F_a , and (c) segment (ν_1, ν^1) occupies the interior of F_b. (d) (μ_2, ν^1) is the intersection of F_a and F_b.

the left side of p_2; $p_1 > p_2$ if p_1 is on the right side of p_2. We thus sort the cuts $(\mu_1, \mu^1, ..., \mu_r, \mu^r)$ and $(\nu_1, \nu^1, ..., \nu_s, \nu^s)$ according to their positions on the cut line $L_{a,b}$; and we can then determine which portion of $L_{a,b}$ forms the intersection of F_a and F_b by the conditions listed below. The segments (μ_i, ν^j) in condition (c1), (ν_j, μ^i) in condition (c2), (μ_i, μ^i) in condition (c3), and (ν_j, ν^j) in condition (c4) all constitute portions of the intersections of F_a with F_b. There is no overlapping between S_a^i and S_b^j in conditions (c5) and (c6).

$$(c1) \quad \nu_j < \mu_i < \nu^j < \mu^i \tag{5.10}$$

$$(c2) \quad \mu_i < \nu_j < \mu^i < \nu^j \tag{5.11}$$

$$(c3) \quad \nu_j < \mu_i < \mu^i < \nu^j \tag{5.12}$$

$$(c4) \quad \mu_i < \nu^j < \nu_j < \mu^i \tag{5.13}$$

$$(c5) \quad \mu_i < \mu^i < \nu_j < \nu^j \tag{5.14}$$

$$(c6) \quad \nu_j < \nu^j < \mu_i < \mu^i \tag{5.15}$$

By sorting the cuts $(\mu_1, \mu^1, ..., \mu_r, \mu^r)$ and $(\nu_1, \nu^1, ..., \nu_s, \nu^s)$ along the cut line $L_{a,b}$, we can select the appropriate segments according to the rules above by means of the following algorithm. The algorithm advances along the cut line and searches for the pair of cuts which constitutes a portion of the intersection of F_a with F_b. It then advances along the cut line to compare each two pairs (μ_i, μ^i) and (ν_j, ν^j) to find the segment which satisfies either one of the following categories: (μ_i, ν^j) in condition (c1), (ν_j, μ^i) in condition (c2), (μ_i, μ^i) in condition (c3), or (ν_j, ν^j) in condition (c4).

(Step 1) Set $i \Leftarrow 1$, and $j \Leftarrow 1$;

(Step 2) If $i <= r$ and $j <= s$ then
compare segments (μ_i, μ^i) with (ν_j, ν^j)
else go to end;

(Step 3) If condition (c1) then select segment (μ_i, ν^j); $j \Leftarrow j + 1$;
If condition (c2) then select segment (ν_j, μ^i); $i \Leftarrow i + 1$;
If condition (c3) then select segment (μ_i, μ^i); $i \Leftarrow i + 1$;
If condition (c4) then select segment (ν_j, ν^j); $j \Leftarrow j + 1$;
If condition (c5) then select no segment; $i \Leftarrow i + 1$;
If condition (c6) then select no segment; $j \Leftarrow j + 1$;

(Step 4) Go to step 2;

(Step 5) End;

5.4 Perturbation

Here we show that the perturbations discussed in sections 2 and 3, $< \epsilon_x, \epsilon_y >$ and $< \epsilon_x, \epsilon_y, \epsilon_z >$ respectively, are physical.

Let $\{V_a^i =< x_a^i, y_a^i >\}$, for $i = 1, ..., I$, be a set of vertices belonging to polygon Q_a that causes singularities on some edges in $\{E_b^j = (V_b^j, V_b^{j+1})\}$, for $j = 1, ..., J$, belonging to polygon Q_b . Let $\{V_b^m =< x_b^m, y_b^m >\}$, for $m = 1, ..., M$, be a set of vertices belonging to polygon Q_b that causes singularities on some edges in $\{E_a^n = (V_a^n, V_a^{n+1})\}$, for $n = 1, ..., N$, belonging to polygon Q_a . Let $\{L_a^n\}$ and $\{L_b^j\}$ be the set of lines which contain edges in $\{E_a^n\}$ and $\{E_b^j\}$, respectively. We assume that the second object Q_b is displaced by an infinitely small perturbation (ϵ_x, ϵ_y) relative to the first object Q_a . Let us represent the line equations of lines in $\{L_a^n\}$ and $\{L_b^j\}$ by

$$\begin{cases} L_a^1 & : \alpha_a^1 x + \beta_a^1 y + \gamma_a^1 = 0 \\ & ..... \\ L_a^N & : \alpha_a^N x + \beta_a^N y + \gamma_a^N = 0 \end{cases} \qquad \begin{cases} L_b^1 & : \alpha_b^1 x + \beta_b^1 y + \gamma_b^1 = 0 \\ & ..... \\ L_b^J & : \alpha_b^J x + \beta_b^J y + \gamma_b^J = 0 \end{cases} \tag{5.16}$$

With an infinitely small perturbation (ϵ_x, ϵ_y) on Q_b , vertices $\{V_b^m =< x_b^m, y_b^m >\}$ are perturbed to new positions $\{V_b^m =< x_b^m + \epsilon_x, y_b^m + \epsilon_y >\}$, so that

$$(V_b^m / L_a^n) = \text{Sign}(\alpha_a^n \epsilon_x + \beta_a^n \epsilon_y)$$

We have assumed that both ϵ_x and ϵ_y are positive and the order of magnitude of ϵ_x is far larger than ϵ_y so that the ϵ_y perturbation on y-axis can be neglected when the ϵ_x perturbation on x-axis is sufficient. Therefore, for those lines in $\{L_a^n\}$ with $\alpha_a^n \neq 0$, the perturbation leads to the results

$$(V_b^m / L_a^n) = \text{Sign}(\alpha_a^n) \quad \text{whenever} \quad \alpha_a^n \neq 0, \tag{5.17}$$

and for those lines in $\{L_a^n\}$ with $\alpha_a^n = 0$, the perturbation leads to the results

$$(V_b^m / L_a^n) = \text{Sign}(\beta_a^n) \quad \text{if} \quad \alpha_a^n = 0 \tag{5.18}$$

What is the constraint on the perturbation $<\epsilon_x, \epsilon_y>$ so that the requirements in (5.17) and (5.18) are fulfilled at the same time? It is easy to tell that the magnitude

of ϵ_y should be restricted to satisfy the condition

$$\text{Sign}(\alpha_a^n \epsilon_x + \beta_a^n \epsilon_y) = \text{Sign}(\alpha_a^n \epsilon_x) \text{ for those lines with } \alpha_a^n \neq 0$$

Therefore, we know that the magnitude of ϵ_y should be small enough to satisfy the condition

$$|\alpha_a^n \epsilon_x| > |\beta_a^n \epsilon_y| \text{ for every } n \text{ where } \alpha_a^n \neq 0$$

This means that the magnitude of ϵ_y should be less than the minimum of $\{\epsilon_x \alpha_a^n / \beta_a^n\}$, for those n with $\alpha_a^n \neq 0$.

A similar rule applies to singularities occurring on vertices $\{V_a^i\}$ of Q_a that coincide with edges $\{E_b^j\}$ of Q_b ; such that

$$\alpha_b^j x_a^i + \beta_b^j y_a^i + \gamma_b^j = 0$$

Polygon Q_b is perturbed by $< \epsilon_x, \epsilon_y >$ relative to polygon Q_a . We interpret this to mean that polygon Q_a is perturbed by $< -\epsilon_x, -\epsilon_y >$ relative to polygon Q_b . After perturbation, vertices $\{V_a^i =< x_a^i, y_a^i >\}$ are displaced to $\{V_a^i =< x_a^i - \epsilon_x, y_a^i - \epsilon_y >\}$ relative to the edges of Q_b . Therefore, we arrive at the result

$$(V_a^i / L_b^j) = \text{Sign}(-\alpha_b^j) \text{ if } \alpha_b^j \neq 0,$$
$$(V_a^i / L_b^j) = \text{Sign}(-\beta_b^j) \text{ if } \alpha_b^j = 0.$$

By the same reasoning, the magnitude of ϵ_y should be restricted to satisfy the condition

$$\text{Sign}(-\alpha_b^j \epsilon_x - \beta_b^j \epsilon_y) = \text{Sign}(-\alpha_b^j \epsilon_x) \text{ for those lines with } \alpha_b^j \neq 0$$

Therefore, we know that the magnitude of ϵ_y should be less than the minimum of $\{\epsilon_x \alpha_b^j / \beta_b^j\}$ for those j with $\alpha_b^j \neq 0$.

We have shown that there does exist such a physical perturbation $<\epsilon_x, \epsilon_y>$ which can fulfill the requirements described above. The constraints on the magnitude of the perturbation are

$$\epsilon_y < \text{minimum}(|\frac{\alpha_a^n}{\beta_a^n}|)\epsilon_x, \text{ for those } n \text{ with } \alpha_a^n \neq 0 \text{ and}$$
$$\epsilon_y < \text{minimum}(|\frac{\alpha_b^j}{\beta_b^j}|)\epsilon_x \text{ for those } j \text{ with } \alpha_b^j \neq 0$$

In performing a set operation on two polyhedra in three-dimensional space, we similarly show that there does exist a physical perturbation $<\epsilon_x, \epsilon_y, \epsilon_z>$ which can transform two polyhedra into a singularity-free situation.

Chapter 6

Skeletal Polyhedron Representation

Highly developed computer representations of sculptured surfaces have been used successfully in computer-aided design in the automotive, aircraft, shipbuilding industries, etc, [3]. However, most of the work has been concentrated on developing elegant mathematical representations for sculptured surfaces instead of addressing the real problem of representing solids. A few researchers who realize the paucity of results in the area of geometric modeling of solids with simple surfaces have looked into various aspects of the problem, including computer-aided design architecture, [10], internal descriptions of objects for computer vision systems, [4, 5], and computer aided design and manufacture of simple mechanical parts, [29].

The objective of the research presented here is to develop a new representation scheme for solids that would lead to a powerful and efficient solid modeling system. To further this goal, we offer a new means of describing solid objects called **Skeletal polyhedron representation** (SPR). It differs significantly from the conventional boundary polyhedron representation scheme which describes solid objects in terms of their boundary faces. A face is a planar polygon bounded by a sequence of edges. An edge is a straight line segment defined by two vertices, and a vertex is a point containing its cartesian coordinates. Since topological information about objects is essentially important in object modeling where general properties such as adjacency and connectedness sometimes provide more useful information than the geometric information, the new SPR representation gives more information about the topology of the object under description.

SPR describes objects by the adjacency and connectedness between vertices. SPR represents a polyhedron by a set of vertices in which a vertex is associated with a radiating-structured bridge which connects the vertex to all its neighbors. The bridges between the vertices constitute the skeleton of a polyhedron; although, the face information of the polyhedron is embedded in the skeletal structure itself. The bridge

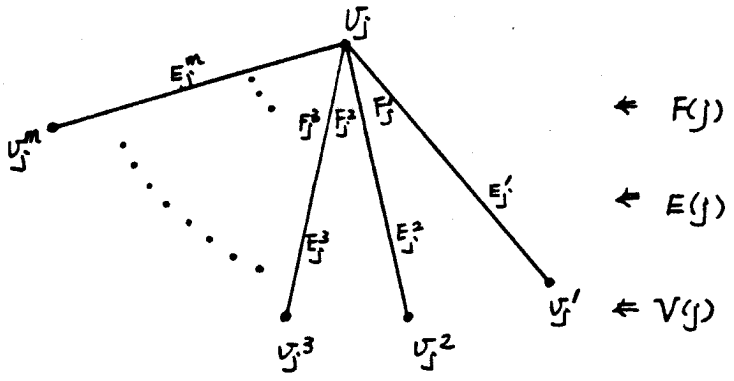

Figure 44

Vertex v_j has a set of neighboring vertices $\{v_j^i\}$, a set of neighboring edges $\{E_j^i\}$, and a set of neighboring faces $\{F_j^i\}$, for $i = 1, .., n$.

structure facilitates set operations on solid objects and also has a great advantage on permitting the creation of volume models from a wire-frame representation. Set operations in SPR can be resolved simply by collecting the appropriate set of vertices and reconstructing the bridge of a vertex locally, unlike the conventional representation where we must recreate boundary faces of the resultant object. SPR can also resolve multiple objects in a set operation. Despite this convenience, SPR contains no less information than conventional boundary polyhedron representation nor than winged-edge polyhedron representation [4].

6.1 Skeletal polyhedron representation

Solids in discussion here are defined as regular sets of points in a Euclidean 3D space. A set is regular if it equals the closure of its interior, [26]. Our discussion is concentrated only on the subset of solids whose boundaries are planar faces, usually called polyhedra. Let Q denote a polyhedron composed of f faces, e edges, and v vertices. Let $\mathcal{F}(Q) = \{F_1, F_2, .., F_f\}$, $\mathcal{E}(Q) = \{E_1, E_2, .., E_e\}$, and $\mathcal{V}(Q) = \{v_1, v_2, .., v_v\}$ denote the set of boundary faces, the set of edges, and the set of vertices of Q, respec-

tively. Each vertex v_j of Q has a set of neighboring vertices, as shown in Figure 44. Let $V(j)$ represent the set of neighboring vertices of vertex v_j, such that

$$V(j) = \{v_k \mid (v_j, v_k) \in \mathcal{E}(Q)\}$$

Also let $E(j)$ represent the set of neighboring edges of v_j which connect the vertex v_j to its neighboring vertices in $V(j)$, such that

$$E(j) = \{(v_j, v_k) \mid (v_j, v_k) \in \mathcal{E}(Q)\}$$

Finally, let $F(j)$ represent the set of faces which converge to the vertex v_j, such that

$$F(j) = \{F_k \mid v_j \in F_k\}$$

The sets $V(j)$, $E(j)$, and $F(j)$ are called, respectively, the *neighboring vertices, neighboring edges* and *neighboring faces* sets of vertex v_j.

The neighboring edges in $E(j)$ surround vertex v_j in a sequential order. Every two consecutive edges expand a corner of a neighboring face of v_j. We can specify the order of a neighboring edge set in either a clockwise or a counter clockwise order. In our representation, we choose to specify a neighboring edge set in a clockwise order surrounding the outward normal vector of the polyhedron.

A representation of a solid in SPR contains a set of vertex-representations, inclusively called a skeletal format. A **vertex-representation** is a description of the relation between a vertex and its neighboring vertices. Let us assume that a vertex v_j has a total of m neighbors. The set of neighboring vertices $V(j)$ of v_j is expressed as a chain $V(j) = (v_j^1, .., v_j^m)$, where v_j^i is called the *pre-neighbor* of v_j^{i+1} and v_j^{i+1} is called the *post-neighbor* of v_j^i. The order of the vertices in $V(j)$ enforces the structure of a polyhedron and also implies that every pair of triple vertices, (v_j^i, v^j, and v_j^{i+1}) constitutes a corner of a neighboring face F_j^i of v_j. The following is the format of a vertex-representation:

$$v_j \quad : \quad v_j^1 \quad \theta_{1,2} \quad v_j^2 \quad \theta_{2,3} \quad .... \quad v_j^m \quad \theta_{jm,1} \qquad \text{where } v_j^k \in V(j) \qquad (6.1)$$

The set of bits $\{\theta_{i,i+1}\}$ in a vertex representation form an *angle-pattern* of the corners of the vertex's neighboring faces. An angle bit $\theta_{i,i+1}$ describes the angle of a

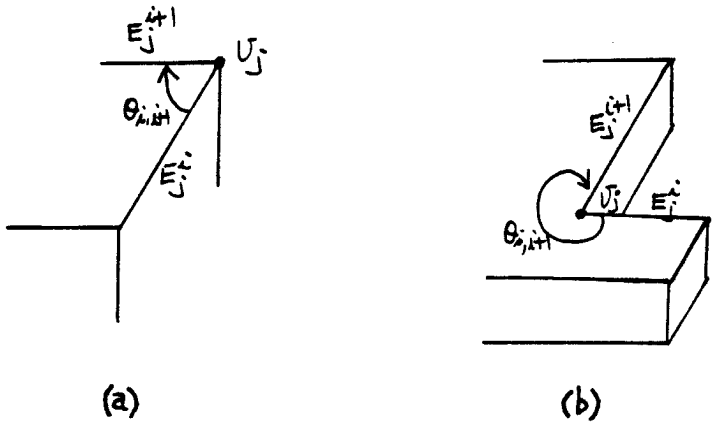

Figure 45
(a) The angle bit $\theta_{i,i+1}$ is less than 180 degrees, (b) the angle bit $\theta_{i,i+1}$ is greater than 180 degrees.

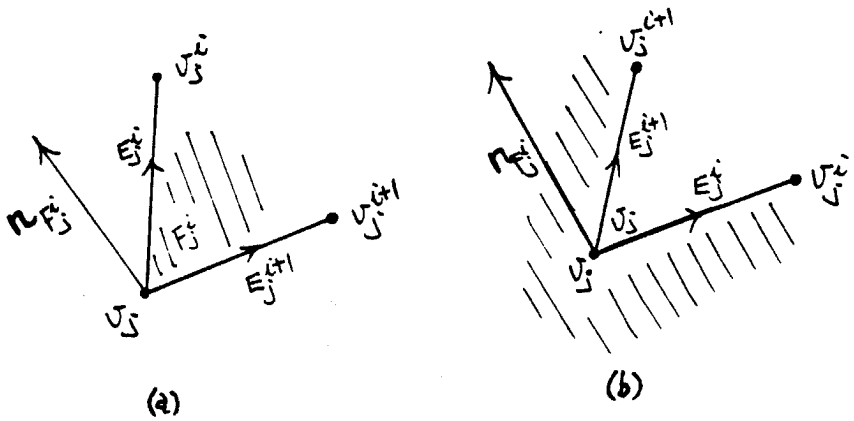

Figure 46
The outward normal vector of face F_j^i is determined by the outer product of edge E_j^i with edge E_j^{i+1} modified by the angle bit $\theta_{i,i+1}$.

corner in between edges $E_j^i = (v_j, v_j^i)$ and $E_j^{i+1} = (v_j, v_j^{i+1})$. An angle bit is defined as either '+' or '–' depending on whether the angle of the corner is greater or less than 180 degrees, as in Figure 45a and 45b.

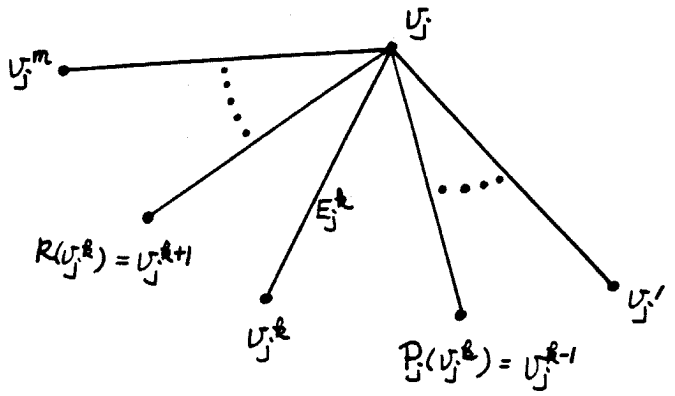

Figure 47

$P_j(v_j^k) = v_j^{k-1}$ is the pre-neighbor of vertex v_j^k, and and $R_j(v_j^k) = v_j^{k+1}$ is the post-neighbor of v_j^k.

$\theta_{i,i+1} = +$ if the corner between edges E_j^i and E_j^{i+1} is less than 180 degrees,
$= -$ if the corner between edges E_j^i and E_j^{i+1} is greater than 180 degrees.

(6.2)

The *outward* normal vector of a face F_j^i can then be determined by the outer product of E_j^i and E_j^{i+1} modified by the angle bit, as in Figure 46.

$$\mathbf{n}_{F_j^i} = \theta_{i,i+1}.(\mathbf{E}_j^i \times \mathbf{E}_j^{i+1}) \tag{6.3}$$

We define two cyclic permutation functions P_j and R_j on $V(j)$ such that for a vertex $v_j^k \in V(j)$, $P_j(v_j^k)$ is the pre-neighbor of v_j^k and $R_j(v_j^k)$ is its post-neighbor, as in Figure 47:

$$P_j : V(j) \Rightarrow V(j)$$
$$R_j : V(j) \Rightarrow V(j)$$

Since both functions P_j and R_j are cyclic, it is obvious that $P_j(v_j^k) = v_j^{k-1}$, $P_j(v_j^1) = v_j^m$, $R_j(v_j^k) = v_j^{k+1}$ and $P_j(v_j^m) = v_j^1$.

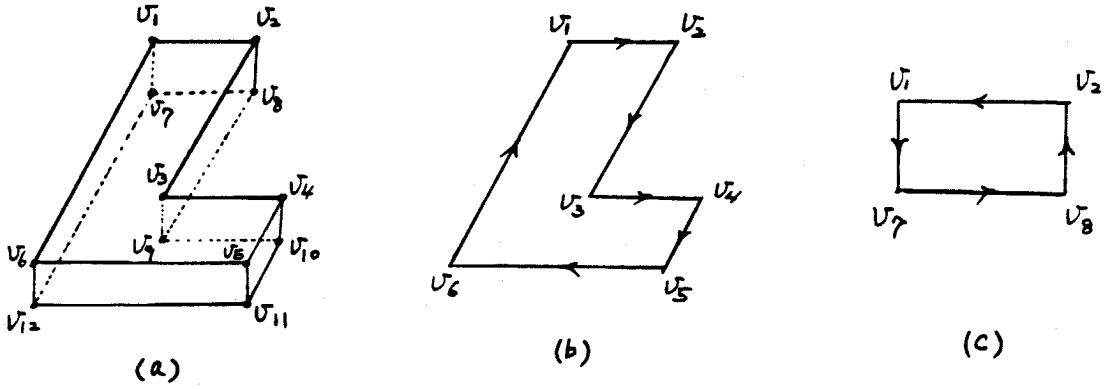

Figure 48

(a) A polyhedron consists of twelves vertices, eighteen edges, and eight faces, (b) a face delineated by polygon $(v_1, v_2, v_3, v_4, v_5, v_8)$, (c) a face delineated by polygon (v_2, v_1, v_7, v_8).

The following is a skeletal representation of an object shown in Figure 48a. To avoid mistaking an angle bit for a plus or a minus sign, in the following representation we denote the '+' angle bit by a period and the '–' angle bit by an asterisk, respectively. We can see from the figure that all corners of the boundary faces are inflexed except two corners, the one on vertex v_3 between edges $E_1 = (v_3, v_2)$ and $E_2 = (v_3, v_4)$ and one on v_9 between edges $E_3 = (v_9, v_8)$ and $E_4 = (v_9, v_{10})$. The third angle bit in the vertex representation of v_3 and the second angle bit in that of v_9 are, therefore, marked by an asterisk.

$$
\begin{array}{lcccccc}
v_1 & : & v_2 & . & v_6 & . & v_7 \; . \\
v_2 & : & v_1 & . & v_8 & . & v_3 \; . \\
v_3 & : & v_2 & . & v_9 & . & v_4 \; * \\
v_4 & : & v_3 & . & v_{10} & . & v_5 \; . \\
v_5 & : & v_4 & . & v_{11} & . & v_6 \; . \\
v_6 & : & v_1 & . & v_5 & . & v_{12} \; . \\
v_7 & : & v_1 & . & v_{12} & . & v_8 \; . \\
v_8 & : & v_2 & . & v_7 & . & v_9 \; . \\
v_9 & : & v_3 & . & v_8 & * & v_{10} \; . \\
v_{10} & : & v_4 & . & v_9 & . & v_{11} \; . \\
v_{11} & : & v_5 & . & v_{10} & . & v_{12} \; . \\
v_{12} & : & v_6 & . & v_{11} & . & v_7 \; .
\end{array}
\tag{6.4}
$$

As the name implies, a skeletal representation characterizes the skeleton of a solid instead of its boundaries, as in boundary representation. The boundary information of a solid in skeletal representation is implicit and is embedded in the bridge structures of the vertices. However, a skeletal representation is different from a wire-frame description, which contains no volume information. Solids represented in SPR are fully and uniquely described.

SPR represents the neighboring vertices of a vertex in a chain so that a boundary face can be easily traced using the *face-tracing* procedure described below. The procedure takes an ordered pair of vertices (v_1, v_2), representing an edge, and searches for the sequence of vertices which delineate the perimeter the face initiated by that edge. The procedure first holds parameter v_1 and then searches under the set of neighboring vertices of the second parameter v_2 to find the pre-neighbor of v_1; if found then it is stored as v_k. The procedure then holds v_2 and searches under the set of neighboring vertices of v_k to find the pre-neighbor of v_2. By recursively following this procedure until v_1 is found again, the polygon initiated by (v_1, v_2) is found.

Here we assume that the edges of a polygon are specified in a clockwise order so that the interior of the polygon is always located to the right side of an edge when the edge of the polygon are traversed.

procedure face-tracing (v_1 , v_2);

```
begin output( 'Polygon=' );
      output(v1);
      vi  :=  v1 ;
      vj  :=  v2 ;
      vk  :=  v2 ;
       while vk ≠ v1  do  begin
      output(vk);
      vk  :=  Pj(vi) ;
      vi  :=  vj ;
      vj  :=  vk ;
       end;
end;
```

For instance, the face shown in Figure 48b can be found by submitting the initiating edge (v_1, v_2) to the face-tracing procedure. The result we get is :

$$\text{Polygon}= v_1, v_2, v_3, v_4, v_5, v_6 \tag{6.5}$$

It is important to notice that the order of the parameters in the procedure is noninvertible. By exchanging the parameters we can arrive at a different result. For instance, by substituting (v_2, v_1) instead of (v_1, v_2) into the face-tracing procedure, we get the polygon

$$\text{Polygon}= v_2, v_1, v_7, v_8 \tag{6.6}$$

This is the face initiated by edge (v_2, v_1), as in Figure 48c.

Occasionally faces of a polyhedron have holes on them, and information on them is also implicit in SPR. Face F of the polyhedron in Figure 49 has a hole. The following is a partially listed skeletal representation of the polyhedron:

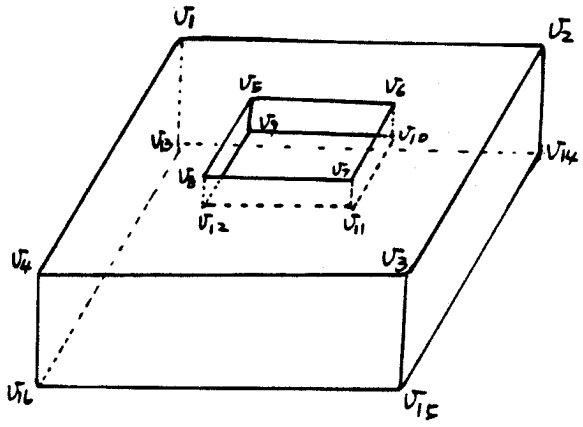

Figure 49

A polyhedron with a hole in one of its faces.

$$
\begin{array}{lcllllll}
v_1 & : & v_2 & . & v_4 & . & v_{13} & . \\
v_2 & : & v_1 & . & v_{14} & . & v_3 & . \\
v_3 & : & v_2 & . & v_{15} & . & v_4 & . \\
v_4 & : & v_1 & . & v_4 & . & v_{16} & . \\
v_5 & : & v_6 & . & v_9 & . & v_8 & * \\
v_6 & : & v_5 & * & v_7 & . & v_{10} & . \\
v_7 & : & v_6 & * & v_8 & . & v_{11} & . \\
v_8 & : & v_5 & . & v_{12} & . & v_7 & * \\
v_9 & : & & .... & & & & \\
.. & : & & .... & & & & \\
v_{16} & : & & .... & & & &
\end{array}
\tag{6.7}
$$

The polygons delineating the perimeter and the hole of face F can be found by substituting $(v_1,\ v_2)$ and $(v_6,\ v_5)$ separately into the parameters of the face-tracing procedure.

$$
\begin{array}{rcl}
\text{perimeter} & : & v_1, v_2, v_3, v_4 \\
\text{hole} & : & v_8, v_7, v_6, v_5
\end{array}
$$

As we can see from the figure, polygon (v_1, v_2, v_3, v_4) delineates the perimeter of F with vertices surrounding the normal vector of F in a clockwise order. An inversion polygon (v_8, v_7, v_6, v_5) delineates a hole in F, but, with vertices surrounding the normal vector in a counter-clockwise order.

6.2 Data structure

An inversion polyhedron is one whose interior occupies the infinity of a 3D space. If we allow no inversion polyhedron to be specified, then the angle-pattern bits in a vertex representation are redundant, see (6.1). A geometrical structure can be unambiguously specified by a set of vertex representations without an angle-pattern, as long as the coordinates of all the vertices are given. However, a corner of a polyhedron described by a vertex representation without an angle-pattern gives no local information on where the interior of the polyhedron resides. A global solution is required to determine the interior region, and sometimes this is very complicated and expensive. The angle-pattern is thus attached so that the solid's regional information can be inferred locally.

As far as implementation is concerned, we prefer a data structure that allows easy access frequently requested information. Next we suggest a data structure for SPR that has the advantage of using no angle-pattern bit information. However, for the purpose of easy access to some data, the data structure must store a certain amount of redundant information. The data structure we suggest is called a redundant skeletal format. A redundant skeletal format dismisses the angle-pattern while still supplying local regional information.

A redundant vertex-representation describes the relation between a vertex and its neighboring edge-set. The neighboring edge-set $E(j)$ of vertex v_j forms a chain structure $E(j) = (E_1^j, .., E_m^j)$, in which E_i^j is the pre-neighbor of E_{i+1}^j (because v_i^j is the pre-neighbor of v_{i+1}^j) and E_{i+1}^j is the post-neighbor of E_i^j. The following is a redundant skeletal format

$$v_j : x, \quad y, \quad z, \quad E_1^j, \quad E_2^j, \quad ..., E_m^j \quad \text{where } E_k^j \in E(j) \tag{6.8}$$

in which x, y, and z are the coordinates of the vertex. Edges E_i^j and E_{i+1}^j constitute a corner of the face between them. The two cyclic permutation functions P_j and R_j can also be defined on $E(j)$, where $P_j(E_k)$ is the pre-neighbor of E_k and $R_j(v_k)$ is the post-neighbor of E_k, such that

$$P_j : E(j) \Rightarrow E(j) \text{ and } R_j : E(j) \Rightarrow E(j)$$

An edge E is a directed line segment defined by an *edge representation.* An edge representation describes the relation between an edge and its four neighbors v_1 and v_2 (the head and the tail of E) and F_r and F_l (the right- and left-hand side consecutive faces of E,) as seen from the exterior along the direction of E. The following is an edge-representation:

$$E \quad : \quad v_1, \quad v_2, \quad F_r, \quad F_l \tag{6.9}$$

A face is a planar polygon, possibly with some holes in it. A face F is described by an outward normal vector $< a, b, c >$, a plane equation $ax + by + cz + d = 0$ of the plane, and an initial edge E_o. A face is defined by a *face representation* as:

$$F \quad : \quad a, \quad b, \quad c, \quad d, \quad e_o, \quad \{h_i\} \tag{6.10}$$

Edge e_o is the lowest indexed edge in the perimeter of F that can be used in the face-tracing procedure to find the entire perimeter of the face. The set of edges in $\{h_i\}$ are initiating edges of holes in F, which can also be used to find the polygons delineating the holes. Since an outward normal vector of a face already implies the side the interior side of a face, the angle pattern can be dismissed in a redundant skeletal format.

The following is a redundant skeletal representation of the solid shown in Figure 48a containing twelves vertices, eighteen edges, and eight faces.

Edge-representation	:	v_1,	v_2,	F_r,	F_l
E_1	:	v_1,	v_2,	F_1,	F_3
E_2	:	v_2,	v_3,	F_1,	F_4
E_3	:	v_3,	v_4,	F_1,	F_5

E_4	:	v_4,	v_5,	F_1,	F_6
E_5	:	v_5,	v_6,	F_1,	F_7
E_6	:	v_6,	v_1,	F_1,	F_8
E_7	:	v_7,	v_8,	F_3,	F_2
E_8	:	v_8,	v_9,	F_4,	F_2
E_9	:	v_9,	v_{10},	F_5,	F_2
E_{10}	:	v_{10},	v_{11},	F_6,	F_2
E_{11}	:	v_{11},	v_{12},	F_7,	F_2
E_{12}	:	v_{12},	v_{13},	F_8,	F_2
E_{13}	:	v_1,	v_7,	F_3,	F_8
E_{14}	:	v_2,	v_8,	F_4,	F_3
E_{15}	:	v_3,	v_9,	F_5,	F_4
E_{16}	:	v_4,	v_{10},	F_6,	F_5
E_{17}	:	v_5,	v_{11},	F_7,	F_6
E_{18}	:	v_6,	v_{12},	F_8,	F_7

Face-represent.	:	a,	b,	c,	d,	eo,	$h_1, ..., h_N$
F_1	:	0	0	1	−1	E_1	
F_2	:	0	0	1	0	E_7	
F_3	:	0	1	0	−2	E_1	
F_4	:	1	0	0	−1	E_2	
F_5	:	0	1	0	−1	E_3	
F_6	:	1	0	0	−2	E_4	
F_7	:	0	1	0	0	E_5	
F_8	:	1	0	0	0	E_6	

Vertex-representation	:	x,	y,	z,	$e1$,	$e2$,	.., en
v_1	:	0	2	1	E_1	E_6	E_{13}

v_2	:	1	2	1	E_2	E_1	E_{14}
v_3	:	1	1	1	E_3	E_2	E_{15}
v_4	:	2	1	1	E_4	E_3	E_{16}
v_5	:	2	0	1	E_5	E_4	E_{17}
v_6	:	0	0	1	E_6	E_5	E_{18}
v_7	:	0	2	0	E_{12}	E_7	E_{13}
v_8	:	1	2	0	E_7	E_8	E_{14}
v_9	:	1	1	0	E_8	E_9	E_{15}
v_{10}	:	2	1	0	E_9	E_{10}	E_{16}
v_{11}	:	2	0	0	E_{10}	E_{11}	E_{17}
v_{12}	:	0	0	0	E_{11}	E_{12}	E_{18}

Compared with the conventional boundary representation, SPR adopts a bottoms-up description structure. In conventional boundary representation, an object is described by a set of bounding faces. Each bounding face is a planar polygon described by a sequence of edges. An edge is then described by two vertices. Finally, a vertex is a point in a Euclidean space which has a set of coordinates associated with it. A vertex is conventionally used as a leaf node that supplies only coordinate information. A face usually sits in a superior position having a normal vector and from there it infers a sequence of vertices that delineate its perimeter. An edge usually connects the faces with the vertices.

In the data structure of skeletal polyhedron representation, a vertex not only has a set of coordinates but also a data structure on all the neighboring vertices, edges and faces attached to it. This data structure gives a full topological description of the vertex. An edge is described by its two ends and two consecutive faces. The wings of an edge in a winged-edge polyhedron representation are part of the information contained in a vertex. A boundary face in this data structure is only implicitly described by a plane equation and an initial edge.

6.3 Discussion

Skeletal polyhedron representation is equivalent to the winged-edge polyhedron

representation [4, 5]. A polyhedron in winged-edge polyhedron representation is made up of four kinds of nodes: bodies, faces, edges, and vertices. The body node is the head of three rings: a ring of faces, a ring of edges, and a ring of vertices. A ring is a doubly-linked circular list with a head node. The perimeter of a face is an ordered list of edges and vertices. The perimeter of a vertex is an ordered list of faces and edges. The perimeter of an edge is an ordered list of two faces and two vertices.

In the data structure of a winged-edge polyhedron representation, each face and each vertex points directly at only one of the edges on its perimeter. Each edge points at its two faces and its two vertices. Furthermore, each edge node contains a link to each of its four immediate neighboring edges clockwise and counter-clockwise about its face perimeter, as seen from the exterior side of the polyhedron. These last four links are the wings of the edge.

Comparing the skeletal representation with the winged-edge representation, we can see that skeletal polyhedron representation offers a more direct description of the topology of an object. It associates a vertex with its neighbors directly, and an edge only with a perimeter consisting of two faces and two vertices. A vertex node takes a leading position in skeletal representation instead of an edge node in winged-edge representation; and the vertex node carries most of the topological information about the object as well, instead of the edge node in winged-edge representation. A face node plays the same role in both representations.

Skeletal polyhedron representation is a complete and unique representation. It is equivalent to conventional boundary representation in that it traces all the boundary faces of a polyhedron with a face-tracing procedure. A redundant skeletal format also supplies information contained in winged-edge representation. Furthermore, SPR has great potential in applications where conventional methods fall short, such as set operations on polygons and solids, the calculation of integral properties, object identification, volume modeling from wire-frame representation, and computer vision. We illustrate the advantages of using a redundant skeletal format in the next chapter where set operations (union, intersection, difference) on polygons and solid objects are discussed.

With regard to the consistency of the representation, an arbitrary permutation of a neighboring set could easily produce a nonphysical solid. However, according to Edmonds' theorem [11], we know that for any connected linear graph with an arbitrary specified cyclic ordering of the edges to each vertex, there always exists a topologically-unique embedding in an oriented closed surface so that the clockwise-edge orderings around each vertex are as specified. The theorem only guarantees the existence of an object satisfying the topological constraints on the edge orderings around each vertex as permuted. However, with geometrical and coordinate constraints, an arbitrary permutation of a neighboring set produces only a nonphysical solid.

Chapter 7

Set Operations on Solids

Computing the union, intersection, or difference of two polygons or two polyhedra is central to operations such as polygon clipping, graphic display, VLSI layout, and other aspects of computer-aided design. A solid modeling system must allow a designer to indicate whether an entity is a fraction of a part or whether that fraction contributes a positive ("solid") or negative ("hole") volume to the complete object. Set operations on solids permit complex objects to be "glued" or "molded" together from simpler shapes or, conversely, to be created by "cutting and drilling" sections out of parts. Techniques such as these greatly reduces the burden of describing the polyhedral representation of complex parts.

A polygon is a planar closed region confined by a sequence of line segments, called edges, that form the boundary of the polygon. In the discussion below we use the notation $E(P)$ to denote the boundary of polygon P which contains a set of edges. The boundaries of two polygons P and Q can be classified into the following sets:

- $\{E(P) - \text{in} - Q\} \Rightarrow$ those portions of edges of P which lie inside Q;

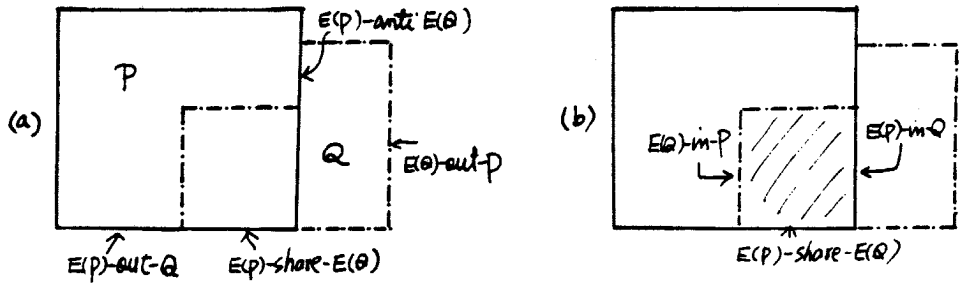

Figure 50

(a) Segments belonging to sets $\{E(P) - \text{out} - Q\}$, $\{E(Q) - \text{out} - P\}$, and $\{E(P) - \text{share} - E(Q)\}$ are selected for $P \cup Q$. (b) segments in sets $\{E(P) - \text{in} - Q\}$, $\{E(Q) - \text{in} - P\}$, and $\{E(P) - \text{share} - E(Q)\}$ are selected for $P \cap Q$.

set	$P \cup Q$	$P \cap Q$	$P - Q$	$Q - P$
$E(P) - \text{in} - Q$		X		X
$E(Q) - \text{in} - P$		X	X	
$E(P) - \text{out} - Q$	X		X	
$E(Q) - \text{out} - P$	X			X
$E(P) - \text{share} - E(Q)$	X	X		
$E(P) - \text{anti} - E(Q)$			X	X

Table 3

- $\{E(Q) - \text{in} - P\} \Rightarrow$ those portions of edges of Q which lie inside P;
- $\{E(P) - \text{out} - Q\} \Rightarrow$ those portions of edges of P which lie outside Q;
- $\{E(Q) - \text{out} - P\} \Rightarrow$ those portions of edges of Q which lie outside P;
- $\{E(P) - \text{share} - E(Q)\} \Rightarrow$ those edges of P and Q which overlap with the interiors occupying same side of the edge; and
- $\{E(P) - \text{anti} - E(Q)\} \Rightarrow$ those edges of P and Q which overlap with the interiors occupying opposite sides of the edge.

Set operations on polygons such as union ($P \cup Q$), intersection ($P \cap Q$) and difference ($P - Q$ or $Q - P$), can be performed via boundary classification - selecting proper boundary fragments out of the sets described above following the rules in Table 3 [12, 23]. The line segments selected must be rearranged in a sequential order.

For instance, to perform a union operation on polygons P and Q, those segments belonging to sets $\{E(P) - \text{out} - Q\}$, $\{E(Q) - \text{out} - P\}$, and $\{E(P) - \text{share} - E(Q)\}$ must be selected, as in Figure 50a. To perform an intersection operation on P and Q, the segments falling in sets $\{E(P) - \text{in} - Q\}$, $\{E(Q) - \text{in} - P\}$, and $\{E(P) - \text{share} - E(Q)\}$ must be selected, as in Figure 50b.

7.1 Skeletal polygon representation

A vertex of a polygon has two neighboring vertices and two neighboring edges. Let vertex v_r be the neighboring vertex to the right of a vertex v and v_l be the

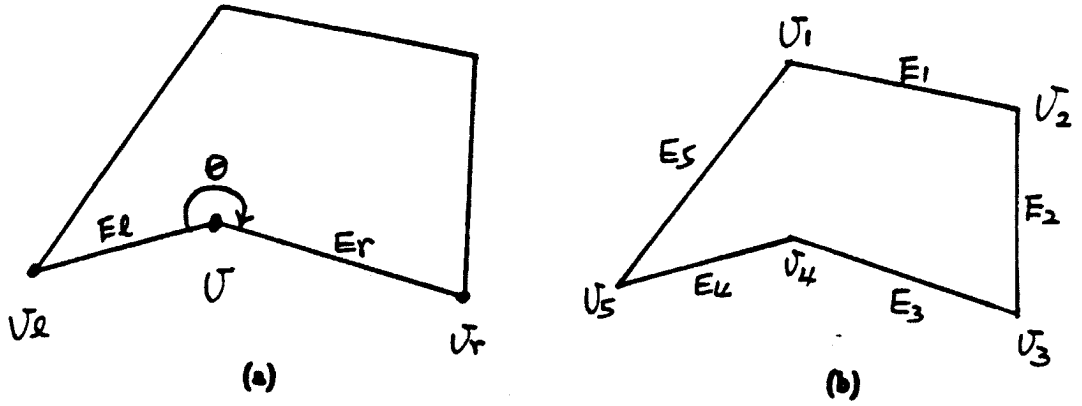

Figure 51

(a) A vertex has two neighboring vertices and two neighboring edges. Vertex v_r and edge E_r are the right-neighbors and vertex v_l and edge E_l are the left-neighbors of v. (b) A polygon represented in a skeletal polygon representation.

neighboring vertex of v to the left. Here 'right' and 'left' are seen from the exterior of the polygon. Let edge E_r denote the neighboring edge to the right of v and E_l the neighboring edge to the left, respectively, as in Figure 51a. The two edges are composed of vertices $E_r = (v_o, v_r)$ and $E_l = (v_o, v_l)$. Usually, we address both E_r and v_r as the *right neighbor* of vertex v and E_l and v_l as the *left neighbor* of v, except in cases where there might be misunderstanding.

We describe a polygon in two-dimensional space by a skeletal polygon representation which consists of a set of 2D vertex-representations. A 2D vertex-representation describes the relationship between a vertex and its left and right neighbors. A vertex-representation can be expressed either as

$$v \quad : \quad v_r \quad {}_{\theta} \quad v_l, \text{ or} \tag{7.1}$$

$$v \quad : \quad E_r, \quad E_l \tag{7.2}$$

The angle bit θ between E_r and E_l, which can be either greater or less than 180 degrees, can be easily determined by the sign resulting from the substitution of the coordinates

of v_r into the line equation of edge E_l. The following is a skeletal representation of a polygon in Figure 51b.

$$\begin{array}{lcll} v_1 & : & E_5, & E_1 \\ v_2 & : & E_1, & E_2 \\ v_3 & : & E_2, & E_3 \\ v_4 & : & E_3, & E_4 \\ v_5 & : & E_4, & E_5 \end{array} \tag{7.3}$$

7.2 Set operations on polygons

The first step in performing a set operation on two polygons is polygon-polygon intersection. This routine compares all edges between two polygons to find all the intersections between them. Polygon-polygon intersection is composed of multiple local operations, called edge-edge intersection, which compare two edges and determine whether they intersect or not. After polygon-polygon intersection, vertices of two polygons are classified into different classes so that one can tell which vertices of the first polygon are located inside (or outside) of the second polygon, and vice versa. By using the skeletal polygon representation, we can easily accomplish a set operation by first collecting the appropriate set of vertices belonging to the resultant polygon and then connecting each vertex to its appropriate neighbors.

A perturbation method presented in Chapter 6 can be used to transform two polygons into singularity-free situations so that each edge-edge intersection can be strictly determined; that is, two edges can only intersect 'normally' or have no intersection. This permits us to classify the boundaries of the two polygons into only four sets rather than six:

- $\{E(P) - \text{in} - Q\} \Rightarrow$ those portions of edges of P which lie inside Q;
- $\{E(Q) - \text{in} - P\} \Rightarrow$ those portions of edges of Q which lie inside P;
- $\{E(P) - \text{out} - Q\} \Rightarrow$ those portions of edges of P which lie outside Q;
- $\{E(Q) - \text{out} - P\} \Rightarrow$ those portions of edges of Q which lie outside P;

The other two sets, $\{E(P) - \text{share} - E(Q)\}$ and $\{E(P) - \text{anti} - E(Q)\}$, present only in singular cases, no longer exist.

The resultant polygon constructed from a set operation on two polygons is called a set-combined polygon. As we know, boundaries of set-combined polygons, $P \cup Q$, $P \cap Q$, $P - Q$, or $Q - P$, are composed of portions of the boundaries of either P or Q. For instance, if polygon W is the union of P and Q, then the boundary of W is composed of those portions of boundaries of P and Q which exterior to the other polygon; if $W = P \cap Q$, then the boundary of W is composed of those portions of boundaries of P and Q which are interior to the other polygon. We can describe these relationships as followings:

$$\text{If } W = P \cup Q, \text{ then } E(W) = \{E(P) - \text{in} - Q\} \cup \{E(Q) - \text{in} - P\} \qquad (7.4)$$

$$\text{If } W = P \cap Q, \text{ then } E(W) = \{E(P) - \text{in} - Q\} \cup \{E(Q) - \text{in} - P\} \qquad (7.5)$$

$$\text{If } W = P - Q, \text{ then } E(W) = \{E(P) - \text{out} - Q\} \cup \{E(Q) - \text{in} - P\} \qquad (7.6)$$

By using the perturbation method, we can strictly classify the vertices of the two polygons into the following four sets. We use $V(P)$ and $V(Q)$ in the following discussion to denote the sets of vertices of polygons P and Q, respectively.

- $\{V(P) - \text{in} - Q\} \Rightarrow$ those vertices of P which lie inside Q;
- $\{V(Q) - \text{in} - P\} \Rightarrow$ those vertices of Q which lie inside P;
- $\{V(P) - \text{out} - Q\} \Rightarrow$ those vertices of P which lie outside Q;
- $\{V(Q) - \text{out} - P\} \Rightarrow$ those vertices of Q which lie outside P.

Two more sets of vertices are possible:

- $\{V(P) - \text{on} - E(Q)\} \Rightarrow$ those vertices of P which lie on boundary of Q;
- $\{V(Q) - \text{on} - E(P)\} \Rightarrow$ those vertices of Q which lie on boundary of P.

However, they are present only in singular situations. The perturbation method resolves this problem by shifting a coincident vertex to one side of the edge so that no vertex is on the boundary of the other polygon.

We can notice that vertices of a set-combined polygon can either be vertices of the two original polygons P and Q, or new vertices created from a normal intersection of two edges of P and Q. We denote this set of new vertices by $\{E(P) - \text{cross} - E(Q)\}$. When a polygon-polygon intersection is performed on P and Q, these new vertices are

set	$P \cup Q$	$P \cap Q$	$P - Q$	$Q - P$
$V(P) - \text{in} - Q$		X		X
$V(Q) - \text{in} - P$		X	X	
$V(P) - \text{out} - Q$	X		X	
$V(Q) - \text{out} - P$	X			X
$E(P) - \text{cross} - E(Q)$	X	X	X	X

Table 4

also created at the same time, and they eventually belong to the resultant polygon. To perform a set operation we then need to collect the appropriate set of vertices belonging to the resultant polygon. The types of vertices collected are dependent on the kind of operation performed. For instance, to perform a union of P and Q, vertices belonging to $\{V(P) - \text{out} - Q\}$, $\{V(Q) - \text{out} - P\}$, and $\{E(P) - \text{cross} - E(Q)\}$ are collected. For the rest of operations $P \cup Q$, $P \cap Q$, $P - Q$, or $Q - P$, we can collect the appropriate set of vertices following the rules in Table 4.

In the next step "bridge construction" is performed to connect each vertex of the resultant polygon to its proper neighbors. The following rules explain how to construct the bridges between a vertex collected for the resultant polygon and its left and right neighbors.

We know from Table 4 that vertices belonging to classes $\{V(P) - \text{out} - Q\}$ or $\{V(Q) - \text{out} - P\}$ are collected only in the operations $P \cup Q$, $P - Q$, and $Q - P$. In performing a union operation on two polygons shown in Figure 52a, we connect vertex v_1 to its two closest neighbors, v_4 and v_{11}, as in Figure 52b, where v_4 becomes the right neighbor and v_{11} becomes the left neighbor of v_1. We express this by the vertex-representation

$$v_1 \quad : \quad v_4, \quad v_{11}$$

This leads to Rule 1: A vertex belonging to classes $\{V(P)-\text{out}-Q\}$ or $\{V(Q)-\text{out}-P\}$ is always connected to the two closest vertices on its two old neighboring edges.

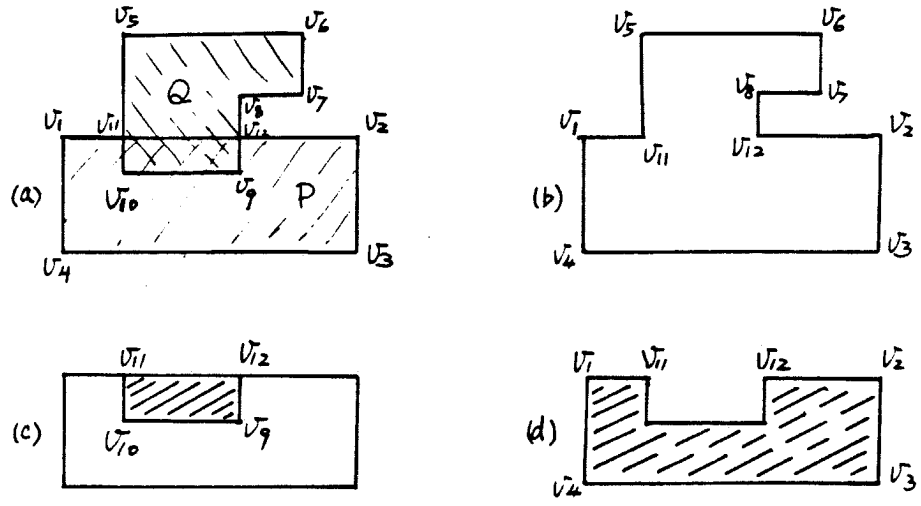

Figure 52
(a) Two polygons $P = (v_1, v_2, v_3, v_4)$ and $Q = (v_5, v_6, v_7, v_8, v_9, v_{10})$.
(b) Union of P and Q. (c) Intersection of P and Q. (d) Subtraction of Q from P.

A vertex of class $\{V(P) - \text{in} - Q\}$ or $\{V(Q) - \text{in} - P\}$ is connected only in the operations $P \cap Q$, $P - Q$, or $Q - P$. Therefore, these bridges are built only in these operations. For example, to perform intersection operation on the same polygons, v_{10} is connected to v_9 and v_{11}, Figure as in 52c. We can express the relation by the vertex-representation

$$v_{10} \quad : \quad v_9 \quad , \quad v_{11}$$

This leads to Rule 2: A vertex of class $\{V(P) - \text{in} - Q\}$ or $\{V(Q) - \text{in} - P\}$ is is also connected to the two closest vertices on its two old neighboring edges.

When we perform a subtraction operation $P - Q$, v_{10} is connected to v_9 and v_{11} in a reverse order, as in Figure 52d

$$v_{10} \quad : \quad v_{11} \quad , \quad v_9$$

Notice that the order of the left and right neighbors are exchanged because the interior is inverted. This occurs also to the vertices of class $\{V(Q)-\text{in}-P\}$ in a $Q-P$ operation.

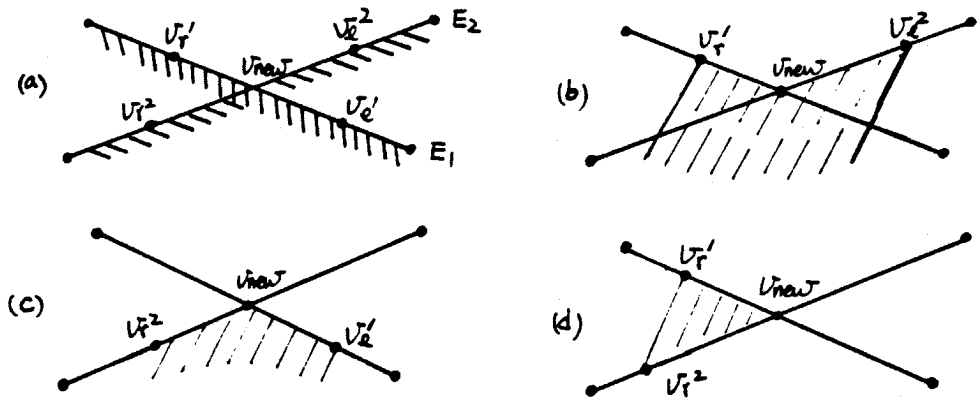

Figure 53

(a) Vertex v_{new} is the intersection of E_1 and E_2. Vertices v_r^1, v_l^1, v_r^2, and v_l^2 are the closest vertices to v_{new} on the four ends of E_1 and E_2. (b) In $P \cup Q$, vertex v_{new} is connected to v_r^1 and v_l^2. (c) In $P \cap Q$, vertex v_{new} is connected to v_r^2 and v_l^1. (d) In $P - Q$, vertex v_{new} is connected to v_r^1 and v_r^2.

Rule 3: A new vertex of class $\{E(P)-\text{cross}-E(Q)\}$ is always collected for every operation, however, it is connected differently depending on the operation performed.

As shown in Figure 53a, E_1 (an edge of P) and E_2 (an edge of Q) intersect by a vertex named v_{new}. The closest vertices to the intersection point v_{new} on the four ends of E_1 and E_2 are denoted by v_r^1, v_l^1, v_r^2, and v_l^2. Vertex v_{new} is connected to v_r^1 and v_l^2 in $P \cup Q$, as shown in Figure 53b, and is connected to v_r^2 and v_l^1 in $P \cap Q$, as in Figure 53c. The relationship of vertex v_{new} to its neighbors is illustrated in Table 5.

The two polygons in Figure 52a illustrate how a union operation on P and Q can be performed. According to Table 4, the vertices collected for $P \cup Q$are $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_{11}$, and $v_{12}\}$, where vertices $v_1, .., v_4$ belong to $\{V(P)-\text{out}-Q\}$, and vertices $v_5, .., v_8$ belong to $\{V(Q)-\text{out}-P\}$. Vertices v_{11} and v_{12} are the newly created vertices. All these vertices are connected to their corresponding neighbors according

operation	vertex representation	illustration
$P \cup Q$	$v_{new} : v_r^1, \quad v_l^2$	Figure 53b
$P \cap Q$	$v_{new} : v_r^2, \quad v_l^1$	Figure 53c
$P - Q$	$v_{new} : v_r^1, \quad v_r^2$	Figure 53d
$Q - P$	$v_{new} : v_l^1, \quad v_l^2$	not shown

Table 5

to the rules explained above. The resultant polygon is shown in Figure 52d and is expressed in a skeletal format as follows:

$$
\begin{array}{lcll}
v_1 & : & v_4, & v_{11} \\
v_2 & : & v_{12}, & v_3 \\
v_3 & : & v_2, & v_4 \\
v_4 & : & v_3, & v_1 \\
v_5 & : & v_{11}, & v_6 \\
v_6 & : & v_5, & v_7 \\
v_7 & : & v_6, & v_8 \\
v_8 & : & v_7, & v_{12} \\
v_{11} & : & v_1, & v_5 \\
v_{12} & : & v_8, & v_2
\end{array}
$$

It is important to point out here that the perturbation method we use has the advantage of being singularity-free; however, it also has the disadvantage of being less flexible. Because perturbing a polygons by an infinitely small displacement $< \epsilon_x, \epsilon_y >$ may create some redundant vertices having the same physical location. As a result, it is possible to create some nonphysical polygons: polygons with zero area. To eliminate this drawback, we delete these redundant vertices later by checking their physical locations.

7.3 Polygon combination algorithm

The method described above for performing polygon combination has been coded and it has performed very well. We outline the key stages of the algorithm. However, the description is designed for easy comprehension. As a result, we outline each stage without describing every detail of the data structures and the algorithms used. Let the two polygons being operated on be named polygon A and polygon B.

(Step 1) *Polygon-polygon intersection*

Compare all edges of polygon A against all edges of polygon B to find all intersections between edges of polygons A and B. A polygon-polygon intersection is composed of multiple independent edge-edge intersection. If one edge of a polygon intersects an edge of the other polygon, two cuts are produced, one on each edge.

(Step 2) *Sorting cuts*

Sort all cuts on each edge which are made from the intersection with all the edges of the opposite polygon. Cuts are sorted according to their positions on the edge.

(Step 3) *Vertices classification*

Classify vertices of one polygon against the other polygon, that is, determine whether a vertex of polygon A is inside or outside B and also whether a vertex of polygon B is inside or outside A. A two-dimensional point-enclosure detection may be required to classify the first vertex of polygon A against polygon B, and the first vertex of polygon B against polygon A. The rest of the vertices can be classified according to the cuts information obtained in the previous step.

(Step 4) *Vertex collection*

Collect proper set of vertices for the resultant polygon according to what type of operation it is in. The rules for vertex collection are described in Table 4.

(Step 5) *Bridge construction*

Connect the vertices collected in step 4 to their corresponding right and left neighbors following the rules described in Table 5. The bridges are constructed depending on the

type of operation being performed.

(Step 6) *Polygon tracing*
After the bridge construction has been finished, the vertex sequence for the resultant polygon is traced.

The distinguishing feature of a skeletal polygon representation is that the topological information of a polygon is localized so that set operations can be performed by reconstructing the bridges of each vertex independently of the other vertices. Unlike the boundary classification method which collects boundary fragments arbitrarily and eventually tries to put them in an order, the order of vertices is already included implicitly in a skeletal representation.

7.4 Set operations on solids

Polyhedra in a three-dimensional space are described by a set of boundary faces. As in a 2D case set operations on two polyhedra P and Q ($P \cup Q$, $P \cap Q$, $P-Q$or $Q-P$) can also be accomplished via boundary classification [12, 23], by classifying the boundaries of the two polyhedra in similar sets as described in the previous section: (1)$\{F(P) - \text{in} - Q\}$, (2) $\{F(Q) - \text{in} - P\}$, (3) $\{F(P) - \text{out} - Q\}$, (4) $\{F(Q) - \text{out} - P\}$, (5) $\{F(P) - \text{share} - F(Q)\}$, and (6) $\{F(P) - \text{anti} - F(Q)\}$. However, here $F(P)$ and $F(Q)$ denote the sets of bounding faces of polyhedra P and Q, respectively. The sets $\{F(P) - \text{share} - F(Q)\}$ and $\{F(P) - \text{anti} - F(Q)\}$ denote respectively those faces of P and Q that overlap with the interiors occupying the same and the opposite sides of the face. Boundaries of $P \cup Q$, $P \cap Q$, $P - Q$, and $Q - P$ can be then formed by selecting proper boundary fragments out of the above sets.

A skeletal polyhedron representation describes a polyhedron by its set of vertices set and by the connection between the vertices and their neighbors. A set operation on two polyhedra can thus be accomplished by selecting the appropriate set of vertices for the resultant polyhedron and connecting each vertex to its corresponding neighbors. The entire operation in a three-dimensional space is similar to that in a two-dimensional space as described in the previous section.

We know that boundary faces of a set-combined polyhedron, $P \cup Q$, $P \cap Q$, $P - Q$, and $Q - P$, are part of the boundaries of either P or Q. However, the edges of a set-combined polyhedron are either edges of P or Qor new edges created from the intersection of faces of P with Q. The vertices of a set-combined polyhedron are either vertices of P or Qor new vertices created from the intersection of edges of P with faces of Q or vice versa. With a polyhedron-polyhedron intersection all intersections between edges and faces of P and Q can be found. Therefore, as a result of polyhedron-polyhedron intersection, new edges are created from the intersection of faces of P with those of Q, and new vertices are created from the intersection of edges of P with faces of Q or edges of Q with faces of P. Besides, as a by-product of polyhedron-polyhedron-intersection, edges of P and Q are chopped into sections which are located either inside or outside the other polyhedron.

Using the perturbation method and the face-face intersection algorithm described in Chapter 7 to perform polyhedron-polyhedron-intersection, we can strictly classify vertices of the two polyhedra into only four sets. We use the notation $V(P)$ and $V(Q)$ to denote the set of vertices of polyhedra P and Q, respectively.

1. $\{V(P) - \text{in} - Q\} \Rightarrow$ those vertices of P which lie inside Q;
2. $\{V(Q) - \text{in} - P\} \Rightarrow$ those vertices of Q which lie inside P;
3. $\{V(P) - \text{out} - Q\} \Rightarrow$ those vertices of P which lie outside Q;
4. $\{V(Q) - \text{out} - P\} \Rightarrow$ those vertices of Q which lie outside P;

Two more sets of vertices can be created from the intersections of edges of one polyhedron with faces of another polyhedron, and vice versa. We use the notation $V(P)$ and $V(Q)$ to denote the set of edges of polyhedra P and Q, respectively.

5. $\{E(P) - \text{cross} - F(Q)\} \Rightarrow$ new vertices created by the intersection of edges of P with faces Q.
6. $\{E(Q) - \text{cross} - F(P)\} \Rightarrow$ new vertices created by the intersection of edges of Q with faces P.

As a result of face-face intersections, edges of the two polyhedra are also chopped into segments which can be classified as being either inside or outside the other polyhedron:

1. $\{E(P) - \text{in} - Q\} \Rightarrow$ those segments of P which lie inside Q;

set	$P \cup Q$	$P \cap Q$	$P - Q$	$Q - P$
$V(P) - \text{in} - Q$		X		X
$V(Q) - \text{in} - P$		X	X	
$V(P) - \text{out} - Q$	X		X	
$V(Q) - \text{out} - P$	X			X
$E(P) - \text{cross} - F(Q)$	X	X	X	X
$E(Q) - \text{cross} - F(P)$	X	X	X	X

Table 6

2. $\{E(Q) - \text{in} - P\} \Rightarrow$ those segments of Q which lie inside P;

3. $\{E(P) - \text{out} - Q\} \Rightarrow$ those segments of P which lie outside Q;

4. $\{E(Q) - \text{out} - P\} \Rightarrow$ those segments of Q which lie outside P;

One more set of edges is created from the intersections of the faces of the two polyhedra.

5. $\{F(P) - \text{cross} - F(Q)\} \Rightarrow$ new edges created by the intersection of faces of P and Q.

As in a 2D case, to perform a set operation on P and Q, we can select vertices for the resultant polyhedron $P \cup Q$, $P \cap Q$, $P-Q$, or $Q-P$ from the six sets of vertices described above, following the rules in Table 6. For instance, vertices belonging to sets $\{V(P)-\text{out}-Q\}$, $\{V(Q)-\text{out}-P\}$ $\{E(P)-\text{cross}-F(Q)\}$, and $\{E(Q)-\text{cross}-F(P)\}$ are selected for $P \cup Q$; vertices belonging to sets $\{V(P) - \text{in} - Q\}$, $\{V(Q) - \text{in} - P\}$ $\{E(P) - \text{cross} - F(Q)\}$, and $\{E(Q) - \text{cross} - F(P)\}$ are selected for $P \cap Q$.

A *bridge construction* then connects each selected vertex to its corresponding neighbors in a proper cyclic order in a manner dependent on the type of operation being performed. A vertex of these classes $\{V(P)-\text{in}-Q\}$, $\{V(Q)-\text{in}-P\}$, $\{V(P)-\text{out}-Q\}$, or $\{V(Q)-\text{out}-P\}$ is always connected to the closest vertices on its original neighboring edges.

A vertex belonging to classes $\{E(P)-\text{cross}-F(Q)\}$ or $\{E(Q)-\text{cross}-F(P)\}$ is also connected to its neighbors in a manner dependent on the type of operation being performed. Let v_{new} denote a new vertex created from the intersection of an edge E

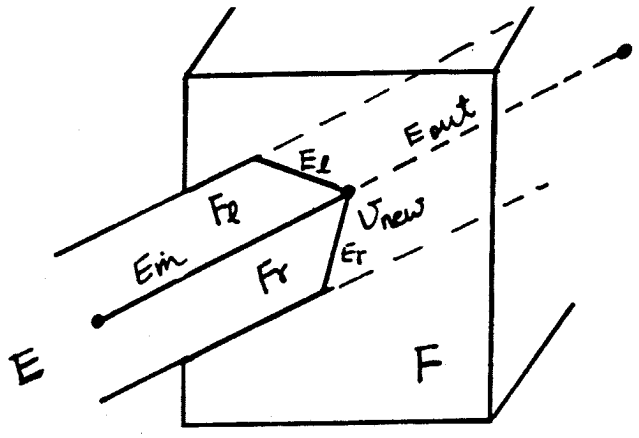

Figure 54

Vertex v_{new} is the intersection of edge E with face F.

of polyhedron P with a face F of polyhedron Q, as in Figure 54. Let E_{in} and E_{out} represent the two segments of E that are near to v_{new}. E_{in} is an edge constituted by v_{new} and the neighboring vertex v_{in} which is located to the interior of F. E_{out} is the edge constituted by v_{new} and the neighboring vertex v_{out} which is located on the exterior side of F. E_{in} and E_{out} are thus two new neighboring edges of v_{new}, one located to the interior and another to the exterior of F. Let F_r and F_l represent the right and left consecutive faces of edge E. Intersections of F_r with F and F_l with F form two edges, denoted by E_r and E_l, that are connected to v_{new}, and are, therefore, new edges of the resultant polyhedron.

For instance, two polyhedra P and Q shown in Figure 55 are composed of vertices $P = \{v_1, v_2, .., v_8\}$ and $Q = \{v_9, v_{10}, .., v_{16}\}$. Four new vertices $\{v_{17}, v_{18}, v_{19}, v_{20}\}$ are created from the intersection of P with Q.

Vertex v_1 belongs to class $\{V(P) - \text{out} - Q\}$, and vertex v_9 belongs to class $\{V(Q) - \text{out} - P\}$. In a union operation $P \cup Q$, vertex v_1 is connected to vertices v_2, v_4, v_{17} and vertex v_9 is connected to vertices v_{10}, v_{12}, v_{13}, the closest vertices on their original neighboring edges, respectively. We can express the relations using the following vertex representations:

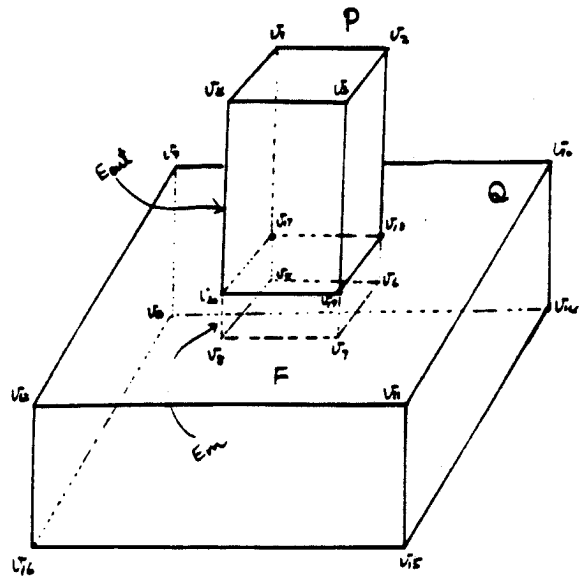

Figure 55

Two polyhedra P and Q are composed of vertices $P = \{v_1, v_2, .., v_8\}$ and $Q = \{v_9, v_{10}, .., v_{16}\}$.

$$v_1 \quad : \quad v_2 \quad . \quad v_4 \quad . \quad v_{17} \quad .$$
$$v_9 \quad : \quad v_{10} \quad . \quad v_{12} \quad . \quad v_{13} \quad .$$

Notice here that vertex v_1 is connected to v_{17} instead of v_5.

Vertex v_5 belongs to class $\{V(P)-\text{in}-Q\}$and is connected to vertices v_8, v_6, v_{17} in an intersection operation $P \cap Q$. Vertices v_8, v_6, v_{17} are the new neighboring vertices of v_5 in $P \cap Q$. We can express these relations using the following vertex representation:

$$v_5 \quad : \quad v_8 \quad . \quad v_6 \quad . \quad v_{17} \quad .$$

Notice again that vertex v_5 is connected to v_{17} instead of v_1.

Vertex v_{20} is a new vertex created from the intersection of an edge $E = (v_4, v_8)$ of polyhedron P with a face $F = (v_9, v_{10}, v_{11}, v_{12})$ of polyhedron Q. Edge $E_{in} = (v_8, v_{20})$ is a new neighboring edge of v_{20} located to the interior of F. $E_{out} = (v_4, v_{20}$ is another new neighboring edge of v_{20} located in the exterior of F. Intersection of right and left consecutive faces of E with F forms two edges $E_1 = (v_{20}, v_{19})$ and

operation	vertex representation
$P \cup Q$	v_{new} : E_r, E_{out}, E_l
$P \cap Q$	v_{new} : E_{in}, E_r, E_l
$P - Q$	v_{new} : E_r, E_{out}, E_l
$Q - P$	v_{new} : E_r, E_{in}, E_l

Table 7

$E_2 = (v_{20}, v_{17})$. In a union operation, vertex v_{20} is connected to vertices v_{17}, v_4, v_{19}, and this can be expressed by a vertex representation

$$v_{20} \quad : \quad v_{17} \quad . \quad v_4 \quad . \quad v_{19} \quad *$$

The rules for constructing the bridges for a new vertex v_{new} created from the intersection of an edge of a polyhedron with a face of another polyhedron are shown in Table 7.

Chapter 8

Basic Engineering Properties Calculation

Volume, center of mass, moments of inertia, and similar properties of solids are defined by triple integrals over subsets of three-dimensional Euclidean space. Because such quantities figure prominently in static and dynamic simulation equations where the mass of an object or the effects during rotation must be calculated prior to manufacture, for instance, the ability to compute integral properties of geometrically complex solids is an important goal in CAD/CAM, robotics, and other fields. The integral properties of a solid, Q, are defined as the volume integral of a function f over the solid:

$$I = \int_Q f(x, y, z)dv.$$

Most computational studies of multiple integrals deal with problems where the domain Q is geometrically simple but the integrand f is complicated. However, in the calculations of mass, moment of inertia, etc., we confront the converse problem: the function f is usually simple but the domain Q may be very complicated.

Lee and Requicha [17,18] have viewed several representation-oriented algorithms for evaluating the triple integral described above. The known methods for representing solids include primitive instancing, disjoint decomposition, simple sweeping constructive solid geometry, and boundary representation. Each of these methods poses particular difficulties in the transfer of theory into practical application.

Objects described by primitive instancing belong to a finite number of object families, each of which is characterized by a finite number of parameters. Algorithms for computing integral properties of objects represented by primitive instancing are primitive-specific: that is, a special formula or method is developed for each primitive. As the number and complexity of primitives in a representation scheme increase so do the programming effort and the size of the software library.

Decomposition methods also suffer from some drawbacks. Disjoint decomposi-

tion partitions a solid into smaller solids. Three-dimensional triangularization decomposes a solid into a union of tetrahedra. Octree decompositions [15] partition objects into cubical solids whose linear dimensions are power-of-two multiple of some minimal size. An integral over a solid is then the sum of integrals over each small solid. However, generating an appropriate decomposition for a solid is usually expensive and requires considerable human labor and computation time.

Likewise, constructive solid geometry (CSG) representation suffers from some relative inefficiencies. In this scheme, objects are described by the union, intersection, and subtraction of primitive solids. A CSG representation is a tree with branching nodes representing operators and leaves representing primitive solids. Lee and Requicha [18] exploit a divide-and-conquer method for computing the integral properties of solids represented by CSG by recursively applying the formulae:

$$\int_{A \cup^* B} f\,dv = \int_A f\,dv + \int_B f\,dv - \int_{A \cap^* B} f\,dv$$

$$\int_{A -^* B} f\,dv = \int_A f\,dv - \int_{A \cap^* B} f\,dv.$$

However, one must solve the basic problem of evaluating an integral over the intersection of an arbitrary number of primitive solids, and this involves extensive computation time.

Other methods, though theoretically exact, may stretch programming skills to their limit. Sweeping representation describes a volume by an object moving along a trajectory, generally with translational and/or rotational motions. Integral properties of solids represented by translational and rotational sweeping may be computed by exploiting dimensional separability to convert a triple integral into a double integral. It remains a difficult challenge to devise a convenient algorithm for this technique, however.

Finally, some methods trade exactness for efficiency. Integral properties of solids represented by boundary representation may be evaluated by generating a collection of quasidisjoint cells whose union approximates the solid, and computing the integral properties of the solid by adding the contribution of each individual. This

method, though simpler, is only approximate and is inefficient. Cohen and Hickey [8] also introduced two algorithms for computing volumes of convex polyhedra. The first algorithm consists of triangularizing a given solid into simplices and adding their individual volumes together. The idea is similar to that contained in this paper. However, our algorithm also is applicable to the calculation of arbitrary polynomial functions. The second algorithm is of the Monte Carlo genre but is specialized to take advantage of the convexity of polyhedra. The result is approximate with the accuracy increased at the expense of additional computer time.

In each of the above methods, objects with complicated boundaries thwart the goals of exactness and ease of execution. By contrast, we present here a simple method for evaluating the integral of an arbitrary polynomial function over an arbitrary possibly nonconvex polyhedron represented by boundary representation.

A direct integral over a polyhedron can be evaluated by taking a central projection and adding the appropriate contributions of the cones defined by the faces of the object with respect to the center of projection [30]. The divergence theorem provides an alternative method for evaluating the integral properties of solids by simply integrating over their boundaries:

$$\int_Q f(x,y,z)dv = \int_Q div(\mathbf{g})dv = \int_{\partial Q} \mathbf{g}.\mathbf{n}ds$$

where $\mathbf{g}$ is a vector function such that the divergence of $\mathbf{g}$ equals the function f, ∂Q is the boundary of Q, $\mathbf{n}$ is the unit outward normal vector of the boundary, and ds is the surface differential.

An integral over a polyhedron can then be calculated easily by using the central projection method and decomposing a polyhedron systematically into a set of simplices and accumulating the results from each simplex based on this formula. We present a general formula for a direct evaluation of the integral of a polynomial over a 3D simplex. This method adopts a systematic and automatic decomposition. The method is analytically exact but the practical accuracy of the result is limited by the accuracy of floating-point arithmetic. The time complexity of this method is linearly proportional to the number of vertices of a polyhedron.

8.1 A symbolic evaluation

The area of a planar region can be defined as a vector which not only has a quantity but also an associated orientation. The area of a triangle T with vertices (v_o, v_1, v_2) equals the outer product of the two vectors $\mathbf{r}_1 = (\mathbf{v}_1 - \mathbf{v}_o)$ and $\mathbf{r}_2 = (\mathbf{v}_2 - \mathbf{v}_o)$. We use $\mathbf{v}$ to denote a vector from the origin to a vertex v.

$$\begin{aligned} Area(T) &= \frac{1}{2}(\mathbf{r}_1 \times \mathbf{r}_2) = \frac{1}{2}(y_1 z_2 - y_2 z_1, z_1 x_2 - x_1 z_2, x_1 y_2 - x_2 y_1), \text{ where} \\ &\mathbf{r}_1 = (x_1, y_1, z_1) \text{ and } \mathbf{r}_2 = (x_2, y_2, z_2). \end{aligned} \tag{8.1}$$

Here the symbol $\times$ denotes the cross-products of two vectors. The area of an arbitrary closed planar region R is:

$$\text{Area}(R) = \frac{1}{2}\oint_{\partial R} \mathbf{r} \times \mathbf{dl}\ , \tag{8.2}$$

where ∂R is the boundary of the region R and $\mathbf{dl}$ is the differential tangent vector of the boundary. The area of a planar polygon $R = (v_1, v_2, .., v_n)$ with n vertices, $\{v_i = (x_i, y_i, z_i)\}$, is:

$$\text{Area}(R) = \frac{1}{2}\oint_{\partial R} \mathbf{r} \times \mathbf{dl} = \frac{1}{2}\sum_{i=1}^{n-2}(\mathbf{v}_{i+1} - \mathbf{v}_1) \times (\mathbf{v}_{i+2} - \mathbf{v}_1). \tag{8.3}$$

Using v_1 as a projection origin, we dissect the planar polygon sequentially into a series of triangles formed by v_1 and each directed edge of the polygon in sequence. The area of the polygon is the vector sum of the areas of each triangle. As we can notice, the dissected triangles are not necessarily mutually disjoint, however, if a polygon is concave, as in Figure 56b. Equation (8.3) nevertheless holds for either convex or nonconvex polygons.

Polygon R_a in Figure 56a is a convex polygon composed of five vertices. Using v_1 as a projection origin, R_a is dissected sequentially into three triangles $\{T_i = (v_1, v_{i+1}, v_{i+2})\}$ for $i = 1, 2, 3$. The area of R_a equals the sum of the areas of each triangle. Polygon R_b in Figure 56b is concave and composed of six vertices. R_b can be dissected into four triangles $\{T_i = (v_1, v_{i+1}, v_{i+2})\}$ for $i = 1, .., 4$, where the triangles $T_1 = (v_1, v_2, v_3)$ and $T_2 = (v_1, v_3, v_4)$ are not disjoint. The area of R_b equals the sum of the appropriately signed areas of each triangle, i.e., $Q_b = T_1 - T_2 + T_3 + T_4$.

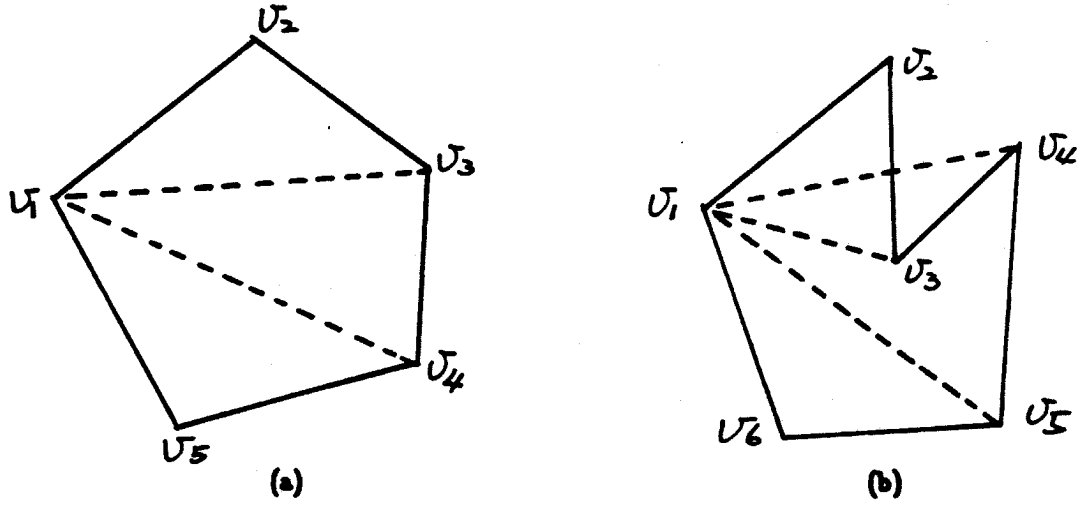

Figure 56

(a) A convex polygon composed of five vertices and dissected into three triangles; (b) a concave polygon composed of six vertices and dissected into four triangles.

The integral properties of a polyhedron Q can be described by:

$$I = \int_Q f(x, y, z)dv, \tag{8.4}$$

where function f is a polynomial. With a linear transformation defined as

$$\begin{cases} x &= g_x(u, v, w) \\ y &= g_y(u, v, w) \\ z &= g_z(u, v, w) \end{cases}$$

equation (8.4) becomes

$$I = \int\int\int_Q f(g_x, g_y, g_z)|J|dudvdw \tag{8.5}$$

where the Jacobian J is

$$J = \begin{vmatrix} \partial g_x/\partial u & \partial g_x/\partial v & \partial g_x/\partial w \\ \partial g_y/\partial u & \partial g_y/\partial v & \partial g_y/\partial w \\ \partial g_z/\partial u & \partial g_z/\partial v & \partial g_z/\partial w \end{vmatrix} \tag{8.6}$$

The integrand f in equation (8.4) is a polynomial which can be generally represented as:

$$f(x,y,z) = \sum_{n_1,n_2,n_3} x^{n_1} y^{n_2} z^{n_3}, \qquad \text{where } n_1, n_2, \text{ and } n_3 \text{ are integers.}$$

To compute the integral in (8.4) we consider only one term at a time:

$$I = \int\int\int_Q x^{n_1} y^{n_2} z^{n_3} dxdydz \tag{8.7}$$

First let us look at a simple case where Q is a 3D simplex, i.e., a tetrahedron, with four vertices (v_o, v_1, v_2, v_3) and the vertex v_o located at the origin. The coordinates of the vertices are:

$$\begin{cases} v_o &= (0,0,0) \\ v_1 &= (x_1, y_1, z_1) \\ v_2 &= (x_2, y_2, z_2) \\ v_3 &= (x_3, y_3, z_3) \end{cases} \tag{8.8}$$

We define a linear transformation T as

$$T = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{pmatrix} \tag{8.9}$$

which relates the old coordinate system (x, y, z) with the new system (X, Y, Z) by

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{8.10}$$

Under this transformation, the tetrahedron $Q = (v_o, v_1, v_2, v_3)$ in (8.8) is transformed into an orthogonal unit tetrahedron $W = (v_o', v_1', v_2', v_3')$ with coordinates:

$$\begin{cases} v_o' &= (0,0,0) \\ v_1' &= (1,0,0) \\ v_2' &= (0,1,0) \\ v_3' &= (0,0,1) \end{cases} \tag{8.11}$$

Based on the transformation in (8.5), the integral in (8.7) becomes

$$\begin{aligned} I &= \int\int\int_Q x^{n_1} y^{n_2} z^{n_3} dxdydz \\ &= \|T\| \int\int\int_W (x_1X + x_2Y + x_3Z)^{n_1} (y_1X + y_2Y + y_3Z)^{n_2} \\ &\qquad \times (z_1X + z_2Y + z_3Z)^{n_3} dXdYdZ \end{aligned} \tag{8.12}$$

where the Jacobian $\|T\|$ equals the absolute value of the determinant of the matrix T.

Next we present a formula for evaluating the integral of a polynomial $x^{n_1} y^{n_2} z^{n_3}$ over an orthogonal unit tetrahedron W described in (8.11):

$$\int_W x^{n_1} y^{n_2} z^{n_3} dv = \int_0^1 \int_0^{1-z} \int_0^{1-z-y} x^{n_1} y^{n_2} z^{n_3} dxdydz = \frac{n_1!\quad n_2!\quad n_3!}{(n_1+n_2+n_3+3)!}. \tag{8.13}$$

The details are shown in the Appendix A. We can see that an arbitrary tetrahedron can always be transformed to an orthogonal unit tetrahedron by means of a simplex-dependent transformation matrix T as in Equation (8.9). Therefore, an integral of a polynomial over a tetrahedron can be evaluated symbolically by Equations (8.12) and (8.13).

To calculate the integral in (8.12), first of all, we decompose the integrand

$$\begin{aligned} I &= \|T\| \int_W (x_1X + x_2Y + x_3Z)^{n_1} (y_1X + y_2Y + y_3Z)^{n_2} (x_1X + z_2Y + z_3Z)^{n_3} dV, \\ &= \|T\| \sum_i \sum_j \sum_k c(i,j,k) \int_W X^i Y^j Z^k dV, \\ &= \|T\| \sum_i \sum_j \sum_k c(i,j,k) \frac{i!\quad j!\quad k!}{(i+j+k+3)!}. \end{aligned} \tag{8.14}$$

Here the function $c(i,j,k)$ represents the coefficient of a term $X^iY^jZ^k$ in the expansion of the integrand, which can be loosely described by

$$\begin{aligned} &(x_1X + x_2Y + x_3Z)^{n_1} (y_1X + y_2Y + y_3Z)^{n_2} (z_1X + z_2Y + z_3Z)^{n_3} \\ &= \sum_{i+j+k=n_1+n_2+n_3} c(i,j,k) X^i Y^j Z^k \end{aligned}$$

The following examples show how to calculate the volume, center of mass, and moments of inertia of a 3D simplex. The volume of a tetrahedron Q as described in (8.8) is

$$V = \int_Q dv = \|T\| \int_W dV = \|T\| \frac{0!}{3!} = \frac{\|T\|}{6} \tag{8.15}$$

The barycenter (x_o, y_o, z_o) of the tetrahedron Q is

$$\begin{aligned} x_o &= \int_Q x dv/V, \\ &= \|T\| \int\int\int_W (x_1 X + x_2 Y + x_3 Z) dX dY dZ/V \\ &= \frac{1}{4}(x_1 + x_2 + x_3) \\ y_o &= \frac{1}{4}(y_1 + y_2 + y_3) \\ z_o &= \frac{1}{4}(z_1 + z_2 + z_3) \end{aligned} \tag{8.16}$$

The moments of inertia of the tetrahedron Q are

$$\begin{aligned} I_{xx} &= \int_Q x^2 dv = \frac{V}{10}(x_1^2 + x_2^2 + x_3^2 + x_1x_2 + x_1x_3 + x_2x_3) \\ I_{yy} &= \int_Q y^2 dv = \frac{V}{10}(y_1^2 + y_2^2 + y_3^2 + y_1y_2 + y_1y_3 + y_2y_3) \\ I_{zz} &= \int_Q z^2 dv = \frac{V}{10}(z_1^2 + z_2^2 + z_3^2 + z_1z_2 + z_1z_3 + z_2z_3) \\ I_{xy} &= \int_Q xy dv \\ &= \frac{V}{20}[2(x_1y_1 + x_2y_2 + x_3y_3) + (x_1y_2 + x_2y_1 + x_1y_3 + x_3y_1 + x_2y_3 + x_3y_2)] \\ I_{yz} &= \frac{V}{20}[2(z_1y_1 + z_2y_2 + z_3y_3) + (z_1y_2 + z_2y_1 + z_1y_3 + z_3y_1 + z_2y_3 + z_3y_2)] \\ I_{xz} &= \frac{V}{20}[2(x_1z_1 + x_2z_2 + x_3z_3) + (x_1z_2 + x_2z_1 + x_1z_3 + x_3z_1 + x_2z_3 + x_3z_2)] \end{aligned} \tag{8.17}$$

Higher order moments of the tetrahedron Q can also be calculated in a similar way.

8.2 Integration over arbitrary nonconvex polyhedra

The integral properties of a solid Q defined in a polar coordinate system can be described as

$$I = \int_Q f(r,\theta,\phi)dv \tag{8.18}$$

Let $\mathbf{G}\ (r,\theta,\phi)$ be a vector function and $g(r,\theta,\phi)$ be a scalar function which is equal to the divergence of $\mathbf{G}$:

$$\begin{aligned} \mathbf{G}\ (r,\theta,\phi) &= g(r,\theta,\phi)\mathbf{r}, \quad \text{and} \\ \nabla\cdot\mathbf{G} &= f \end{aligned}$$

where $\mathbf{r}$ is the unit radial vector. Functions $g(r,\theta,\phi)$ and f are therefore related such that

$$\begin{aligned} f &= \frac{\partial g(r,\theta,\phi)}{\partial r} + 2\frac{g(r,\theta,\phi)}{r} \\ g &= \frac{1}{r^2}\int_0^r r'^2 f(r',\theta,\phi)dr' \end{aligned}$$

Through the divergence theorem, the integral in (8.18) becomes

$$\begin{aligned} I &= \int_Q f(r,\theta,\phi)dv \\ &= \int_Q \nabla\cdot\mathbf{G}\ dv \\ &= \int_{\partial Q} \mathbf{G}\cdot d\mathbf{s} \\ &= \int_{\partial Q} g(r,\theta,\phi)\mathbf{r}\cdot\mathbf{n}ds \end{aligned} \tag{8.19}$$

Therefore, the integral can be represented as a surface integral over the boundary of the polyhedron.

For example, in Figure 57 the integral is taken over a small volume Δv , a cone, which is expanded by a small face Δs of the polyhedron with respect to the origin

$$\int_{\Delta v} f(r,\theta,\phi)dv = \int_{\partial\Delta v} g(r,\theta,\phi)\mathbf{r}\cdot\mathbf{n}ds \tag{8.20}$$

Since the outward normal vector $\mathbf{n}$ of the wall of the cone is orthogonal to the radial vector $\mathbf{r}$, i.e., $\mathbf{r}\cdot\mathbf{n} = 0$, the integral in (8.20) is valuable only on the face Δs

$$\int_{\Delta v} f(r,\theta,\phi)dv = \int_{\Delta s} g(r,\theta,\phi)\mathbf{r}\cdot\mathbf{n}ds \tag{8.21}$$

Let us call the face that expands a cone the *base* of the cone. A volume integral over a cone can, therefore, be represented in terms of a surface integral over the base of the

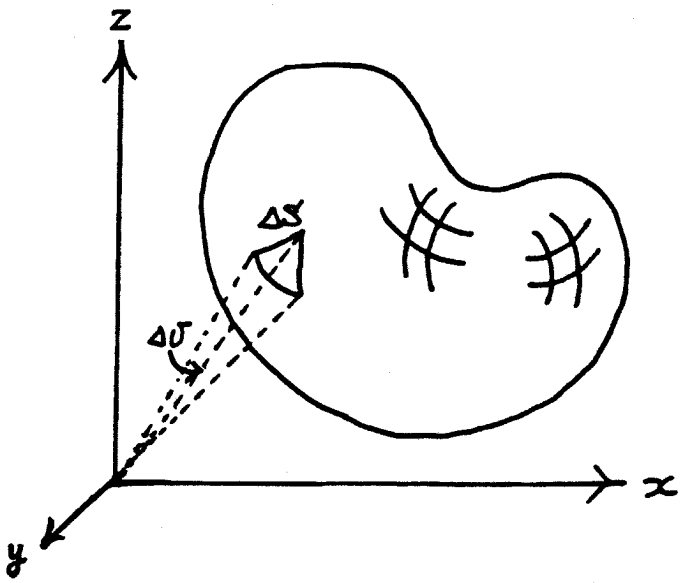

Figure 57

A small cone Δv is expanded by a small face Δs of the polyhedron with respect to the origin. The normal vector on the wall of the cone is orthogonal to the radial vector. A surface integral taken over the surface of the cone, therefore, is valuable only on the face Δs .

cone. The integral in (8.19) thus equals a sum of appropriately signed volume integrals over cones expanded by faces of Q such that

$$\begin{aligned} I &= \int_Q f(r,\theta,\phi)dv \\ &= \int_{\partial Q} g(r,\theta,\phi)\mathbf{r}\cdot\mathbf{n}ds \\ &= \sum_{\Delta v_i} S(\Delta v_i)\int_{\Delta v_i} f(r,\theta,\phi)dv \end{aligned} \tag{8.22}$$

We introduced a *sign function* $S(\Delta v_i)$ in the above equation. The sign function is defined on a cone Δv_i as:

$$\begin{aligned} S(\Delta v_i) &= +1 \quad \text{if } \Delta v_i \text{ is coherent with } Q\ , \\ &= -1 \quad \text{otherwise.} \end{aligned}$$

Significantly, the $\mathbf{n}$ in (8.19) is the outward normal vector of the boundary of the polyhedron Q. However, the $\mathbf{n}$ in (8.21) is the outward normal vector of the boundary

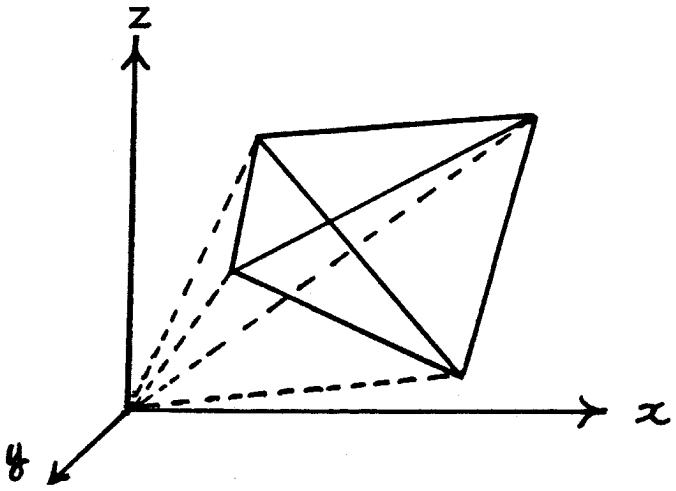

Figure 58
Faces of a tetrahedron expand into four new tetrahedra with respect to the origin.

of the cone Δv_i . They may point toward opposite directions. They point to the same direction only when Δv_i and Q are coherent, i.e., the interiors of Q and Δv_i occupy the same halfspace divided by the base of the cone, Δs_i . The sign function $S(\Delta v_i)$ determines whether the integral over Δv_i has a positive or negative contribution to the integral over Q . If Δv_i is coherent with Q , the integral over Δv_i has a positive contribution to Q ; otherwise it has a negative contribution.

A tetrahedron $Q = (v_1, v_2, v_3, v_4)$ in Figure 58 has four faces, $F_1 = (v_2, v_4, v_3)$, $F_2 = (v_1, v_3, v_4)$, $F_3 = (v_1, v_4, v_2)$, and $F_4 = (v_1, v_2, v_3)$. The order of the vertices in each face are specified clockwise so that the normal vector of a face always points away from the tetrahedron. A face F_i of Q with the origin O forms a new tetrahedron $Q_i = (O, v_m, v_j, v_k)$, where (v_m, v_j, v_k) are vertices of F_i . Four new tetrahedra, $Q_1 = (O, v_2, v_3, v_4)$, $Q_2 = (O, v_1, v_3, v_4)$, $Q_3 = (O, v_1, v_2, v_4)$, and $Q_4 = (O, v_1, v_2, v_3)$ are formed. As we can tell, tetrahedra Q_1,Q_2, and Q_3 are coherent but Q_4 is incoherent with Q . Since each tetrahedron has a vertex located at the origin, an integral over these tetrahedra can be evaluated symbolically with the method illustrated in the last section.

An integral over the tetrahedron Q thus equals the sum of the appropriately signed integral over each newly formed tetrahedron Q_i

$$\begin{aligned} I &= \int_Q f(x,y,z)dv \\ &= \sum_{Q_1,Q_2,Q_3,Q_4} S(Q_i) \int_{Q_i} f(x,y,z)dv \end{aligned} \tag{8.23}$$

As we know the sign functions are $S(Q_1) = S(Q_2) = S(Q_3) = 1$, and $S(Q_4) = -1$. Let the coordinates of the vertices of a face $F_i = (v_m, v_j, v_k)$ be

$$\begin{cases} v_m & = (x_m, y_m, z_m) \\ v_j & = (x_j, y_j, z_j) \\ v_k & = (x_k, y_k, z_k) \end{cases}$$

A linear transformation T_i can be defined as

$$T_i = \begin{pmatrix} x_m & x_j & x_k \\ y_m & y_j & y_k \\ z_m & z_j & z_k \end{pmatrix} \tag{8.24}$$

As we know, the determinant of the transformation matrix is positive if Q_i is coherent with the base F_i , i.e., $|T_i| > 0$; if Q_i is incoherent with F_i , the determinant is negative. The determinant $|T_i|$ implicitly represents the effect of the sign function $S(Q_i)$ and the Jacobian $\|T_i\|$. Equation (8.23) thus becomes

$$\begin{aligned} I &= \int_Q f(x,y,z)dv \\ &= \sum_{Q_i} S(Q_i) \int_{Q_i} f(x,y,z)dv \\ &= \sum_{Q_i} S(Q_i)\|T_i\| \int_{W_i} f(g_x, g_y, g_z)dV \\ &= \sum_{Q_i} |T_i| \int_{W_i} f(g_x, g_y, g_z)dV \end{aligned} \tag{8.25}$$

where functions g_x, g_y,and g_z are defined as

$$\begin{cases} x & = g_x(X,Y,Z) & = x_mX + x_jY + x_kZ \\ y & = g_y(X,Y,Z) & = y_mX + y_jY + y_kZ \\ z & = g_z(X,Y,Z) & = z_mX + z_jY + z_kZ \end{cases}$$

A polyhedron Q composed of f faces, e edges and v vertices can be represented by $Q = (F, E, V)$, where $F = (F_1, .., F_f)$ is a set of faces, $E = (E_1, .., E_e)$ is a set of edges, and $V = (v_1, .., v_v)$ is a set of vertices. A face F_i is composed of f_i vertices, $F_i = (v_1^i, .., v_{f_i}^i)$, where a vertex v_j^iin$V$ and the vertices are specified in an order such that the normal vector of F_i points away from the polyhedron.

A face can also be a nonconvex polygon. As described in the second section, a face F composed of f vertices can be dissected sequentially into $f-2$ triangles. A face $F_i = \{v_1^i, v_2^i, .., v_{f_i}^i\}$ can be dissected into $f_i - 2$ triangles, $\{T_j^i = (v_1^i, v_{j+1}^i, v_{j+2}^i)\}$, by using the vertex v_1^i as the projection origin. A triangle T_j^i with the origin O forms a tetrahedron $Q_j^i = (O, v_1^i, v_{j+1}^i, v_{j+2}^i)$. A face F_i with respect to the origin, therefore, expands a set of tetrahedra $\{Q_j^i\}$. An integral over a cone Y_i which is expanded by the face F_i with respect to the origin thus equals the sum of appropriately signed integrals over all tetrahedra $\{Q_j^i\}$. Let the coordinates of the vertices of $Q_j^i = (O, v_1^i, v_{j+1}^i, v_{j+2}^i)$ be

$$\begin{cases} O & = (0,0,0) \\ v_1^i & = (x_1^i, y_1^i, z_1^i) \\ v_{j+1}^i & = (x_{j+1}^i, y_{j+1}^i, z_{j+1}^i) \\ v_{j+2}^i & = (x_{j+2}^i, y_{j+2}^i, z_{j+2}^i) \end{cases}$$

As described before, a linear transformation, T_j^i, can then be defined as

$$T_j^i = \begin{pmatrix} x_1^i & x_{j+1}^i & x_{j+2}^i \\ y_1^i & y_{j+1}^i & y_{j+2}^i \\ z_1^i & z_{j+1}^i & z_{j+2}^i \end{pmatrix} \tag{8.26}$$

An integral over a cone Y_i expanded by a face F_i equals the sum of integrals over all tetrahedra $\{Q_j^i\}$

$$\begin{aligned} I &= \int_{Y_i} f(x,y,z)dv \\ &= \sum_{Q_j^i} S(Q_j^i) \int_{Q_j^i} f(x,y,z)dv \\ &= \sum_{W_j^i} |T_j^i| \int_{W_j^i} f(g_x, g_y, g_z)dV \end{aligned} \tag{8.27}$$

where W_j^i is an orthogonal unit tetrahedron transformed from Q_j^i and $|T_j^i|$ is the determinant of the matrix T_j^i. Notice again that the sign function $S(Q_j^i)$ determines the contribution of the integral over Q_j^i to the integral over the cone, and the determinant $|T_j^i|$ combines effects of both the sign function and the Jacobian.

Finally, an integral over a polyhedron $Q = (F, E, V)$ equals the sum of the integrals over all cones $\{Y_i\}$ expanded by faces of the polyhedron with respect to the origin O

$$\begin{aligned} I &= \int_Q f(x,y,z)dv \\ &= \sum_{F_i} \int_{Y_i} f(x,y,z)dv \\ &= \sum_{F_i} \sum_{Q_j^i} \int_{Q_j^i} f(x,y,z)dv \\ &= \sum_{F_i} \sum_{W_j^i} |T_j^i| \int_{W_j^i} f(g_x, g_y, g_z)dV \end{aligned} \tag{8.28}$$

The functions g_x, g_y, and g_z are dependent on the simplex Q_j^i and are defined as

$$\begin{cases} x &= g_x(X,Y,Z) &= x_1^i X + x_{j+1}^i Y + x_{j+2}^i Z \\ y &= g_y(X,Y,Z) &= y_1^i X + y_{j+1}^i Y + y_{j+2}^i Z \\ z &= g_z(X,Y,Z) &= z_1^i X + z_{j+1}^i Y + z_{j+2}^i Z \end{cases}$$

8.3 Complexity analysis

The method described above is simple and systematic. The procedure of computation is divided into integrals over a set of cones formed by faces of the polyhedron with respect to the origin. An integral over a cone is then divided into integrals over a set of tetrahedra formed by the triangles dissected from the face with respect to the origin. Although equations (8.13), (8.12), (8.27), and (8.28) look complicated, the basic idea is very simple. An integral over a polyhedron $Q = (F, E, V)$ equals the sum of the appropriately signed integrals over cones formed by faces of Q with respect to the origin, as in (8.28). An integral over a cone equals the sum of the appropriately signed integrals over a set of dissected tetrahedra, as in (8.27).

Since a face F_i with e_i edges is dissected sequentially into $(e_i - 2)$ triangles, the operation takes a total of $\sum(e_i - 2)$ computations. Since an edge is counted twice during the whole scanning procedure, we arrive at

$$\sum_{F_i}(e_i - 2) = 2E - 2F$$

where E is the total number of edges and F is the total number of faces of the polyhedron. According to the Euler Equation $(V - E + F = 2)$ the total number of computations equals $2(V - 2)$, i.e.

$$\sum_{F_i}(e_i - 2) = 2(V - 2)$$

Therefore, the time complexity of the method is linearly proportional to V, the number of vertices of the polyhedron.

It is worthwhile to note that the numerical accuracy of this method can be improved by shifting the origin of the coordinate system to the barycenter of the object, especially to prevent the occurrence of long, thin tetrahedra when an object is far removed from the origin. By using a more sophisticated 3D trianglarization, the efficiency of the algorithm can be improved. It is also important to notice that the computation time of the method does not increase linearly with the number of integral quantities computed, since only one matrix determinant $|T|$ needs to be computed when several integral quantities are calculated at the same time.

This method can be generalized and applied to solids in higher-dimensional space. We discuss this in Chapter 10.

Chapter 9

Integral over a Set-Combined Polyhedron

A set-combined polyhedron is a polyhedron derived from a set operation on two polyhedra, such as a union, intersection, or subtraction operation. This type of solid figures greatly into the field of constructive solid geometry, in which objects are synthesized by means of set operations on primitive solids. This chapter presents a simple method for calculating the integral properties of this type of solids.

9.1 Basic discussion

A symbolic method for evaluating the integral of a polynomial function over a nonconvex polyhedron was presented in chapter 8. In the following discussion we illustrate how the original method can be generalized to solve the more complicated problem of calculating the integral properties of a set-combined polyhedron.

We showed in chapter 8 that a volume integral over a polyhedron Q can be decomposed into integrals over a set of cones $\{Q_i\}$, defined by the boundary faces $\{F_i\}$ of the polyhedron with respect to the origin.

$$\begin{aligned} &\int_Q f(x,y,z)dv \\ &= \sum_{F_i} \int_{Q_i} f(x,y,z)dv \end{aligned} \tag{9.1}$$

A face of a polyhedron can be further decomposed into a set of triangles with respect to a fixed point on the face, where a triangle T_j on a face F is formed by an edge E_j of F relative to a fixed vertex v_o of F. An integral over a cone thus equals the sum of appropriately signed integrals over all the tetrahedra formed by the set of triangles on the face with respect to the origin. In the following equation a tetrahedron Q_j (defined by an edge E_j=(v_j, v_{j+1}), a fixed vertex v_o of the face, and the origin) is transformed into a unit tetrahedron W_j.

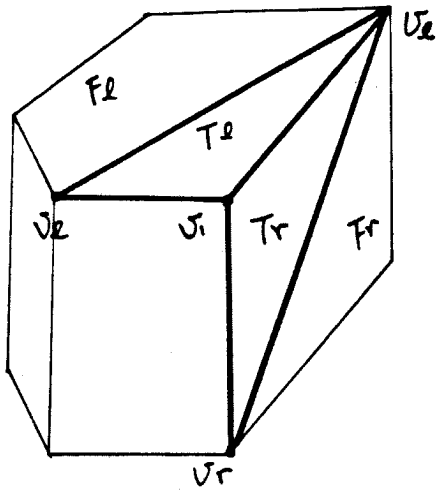

Figure 59

The triangle T_r is constituted by edge E and vertex v_r; T_l is constituted by edge E and vertex v_l.

$$\begin{aligned} &\int_F f(x,y,z)dv \\ &= \sum_{E_j} \int_{Q_j} f(x,y,z)dv, \\ &= \sum_{E_j} |T_j| \int_{W_j} f(x,y,z)dV, \end{aligned} \tag{9.2}$$

The tetrahedron Q_j is transformed into a unit tetrahedron W_j through a transformation T_j defined as

$$T_j = \begin{pmatrix} x_o & x_j & x_{j+1} \\ y_o & y_j & y_{j+1} \\ z_o & z_j & z_{j+1} \end{pmatrix} \quad \text{where} \begin{cases} v_o = (x_o, y_o, z_o) \\ v_j = (x_j, y_j, z_j) \\ v_{j+1} = (x_{j+1}, y_{j+1}, z_{j+1}) \end{cases} \tag{9.3}$$

An integral over a polyhedron thus becomes

$$\int_Q f(x,y,z)dv = \sum_{F_i} \sum_{E_j^i} |T_j^i| \int_{W_j^i} f(x,y,z)dV \tag{9.4}$$

An edge is defined as $E=(v_1, v_2, F_r, F_l)$, where v_1 and v_2 are its head and tail, and F_r and F_l are its right- and left-consecutive faces. One edge constitutes a

pair of triangles, each with respect to a fixed vertex on its two consecutive faces. For instance, the triangle T_r in Figure 59 is constituted by edge E and a fixed vertex v_r of its right-consecutive face F_r, and T_l is constituted by E and a fixed vertex v_l of its left-consecutive face F_l. We can choose the highest indexed vertex of the face as the projection center, for instance, so that all the triangles of a face are formed by the edges of the face with respect to the same vertex.

We can rearrange the order of summation in Equation (9.4) so that it follows the sequence of edges instead of the sequence of faces. Therefore, instead of scanning first through all faces of the polyhedron and then through all edges of each face, we perform the summation by scanning through all edges of the polyhedron where an edge $E = (v_1, v_2)$ constitutes a pair of tetrahedra W_r, W_l, with respect to the origin, and a fixed vertex of each of its consecutive faces. Tetrahedron W_r is constituted by E, the origin, and a fixed vertex v_r on its right consecutive face, while W_l is constituted by E, the origin, and a fixed vertex v_l on its left consecutive face.

$$\begin{aligned} I &= \int_Q f(x,y,z)dv \\ &= \sum_E \Big(\quad |T_r| \int_{W_r} f(f_{rx}, f_{ry}, f_{rz})dV + |T_l| \int_{W_l} f(f_{lx}, f_{ly}, f_{lz})dV \quad \Big) \end{aligned} \tag{9.5}$$

We can define the two polyhedra by the expressions $W_r = (O, v_1, v_2, v_r)$ and $W_l = (O, v_2, v_1, v_l)$. The transformations T_r and T_r are defined by the coordinates of the vertices of the tetrahedra W_r and W_l such that

$$\begin{cases} v_r =< x_r, y_r, z_r > \\ v_l =< x_l, y_l, z_l > \\ v_1 =< x_1, y_1, z_1 > \\ v_2 =< x_2, y_2, z_2 > \end{cases}$$

$$T_r = \begin{pmatrix} x_r & x_1 & x_2 \\ y_r & y_1 & y_2 \\ z_r & z_1 & z_2 \end{pmatrix} \qquad T_l = \begin{pmatrix} x_l & x_2 & x_1 \\ y_l & y_2 & y_1 \\ z_l & z_2 & z_1 \end{pmatrix} \tag{9.6}$$

The functions f_{rx}, f_{ry}, f_{rz}, f_{lx}, f_{ly}, and f_{lz} are defined separately as

$$\begin{pmatrix} f_{rx} \\ f_{ry} \\ f_{rz} \end{pmatrix} = (T_r)\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} f_{lx} \\ f_{ly} \\ f_{lz} \end{pmatrix} = (T_l)\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

We can see from the above analysis that, to compute the integral, using only edge information is sufficient for the calculation. We include an algorithm in Appendix B for calculating the volume, center of mass, and moments of inertia of solids. We can make use of this tool in calculating an integral over a set-combined polyhedron.

Usually, we calculate the integral properties of a set-combined polyhedron by first building the polyhedron using the set operator described in the previous chapter, and then performing an integration by the method described previously. However, we can improve on this approach if we can do the calculation without rebuilding the polyhedron. In the discussion that follows we offer a method for accomplishing this goal.

9.2 Outline of the method

Let P and Q denote two polyhedra, and W a set-combined polyhedron of P and Q. We will use $F(P)$, $F(Q)$ and $F(W)$ to denote the boundaries of the polyhedra P, Q, and W, respectively. These boundaries can be expressed as sets of the bounding faces of the polyhedra. We denote the boundary faces of the polyhedra P, Q, and W, respectively, by the expressions $F(P) = \{F_P^i\}$, $F(Q) = \{F_Q^i\}$, and $F(W) = \{F_W^i\}$, where F_P^i, F_Q^i, and F_W^i refer to the i_{th} face of the polyhedra P , Q , and W, respectively.

If $E(F)$ denotes the boundary of a face F , then $E(F) = \{E_i\}$ is a set of edges delineating the perimeter of F , where E_j denotes the j_{th} edge of face F. Therefore, edges of F_P^i, F_Q^i, and F_W^i, respectively, can be expressed as

$$E(F_P^i) = \{E_P^{i,j}\}, \quad E(F_Q^i) = \{E_Q^{i,j}\}, \quad \text{and } E(F_W^i) = \{E_W^{i,j}\}$$

where $E_{i,j}^P$ denotes the j^{th} edge of the i^{th} face of the polyhedron P . We then describe the sets of edges of the polyhedra P, Q, and W by the expressions $E(P)$, $E(Q)$, and $E(W)$, respectively, as

$E(P)=\cup\ \{\partial F_i^P\} =\cup\ \{E_P^{i,j}\}$, $E(Q)=\cup\ \{\partial F_i^Q\} =\cup\ \{E_Q^{i,j}\}$, and $E(W)=\cup\ \{\partial F_i^W\} =\cup\ \{E_W^{i,j}\}$

We can generally describe the relationship between the boundary of a set-combined polyhedron and the boundary of its source polyhedra by the following statements:

$$\text{If } W = P \cup Q, \text{ then } F(W) = \{F(P) \cap \overline{Q}\} + \{F(Q) \cap \overline{P}\} \tag{9.7}$$

$$\text{If } W = P \cap Q, \text{ then } F(W) = \{F(P) \cap Q\} + \{F(Q) \cap P\} \tag{9.8}$$

$$\text{If } W = P - Q, \text{ then } F(W) = \{F(P) \cap \overline{Q}\} + \{F(Q) \cap P\} \tag{9.9}$$

Here we use $F(P) \cap Q$ to denote the portions of bounding faces of polyhedron P which are located in the interior of Q , and $F(P) \cap \overline{Q}$ to denote the portions of bounding faces of polyhedron P which are located outside of Q . Also we use $F(Q) \cap P$ to denote the portions of bounding faces of polyhedron Q which are located in the interior of P, and $F(Q) \cap \overline{P}$ to denote the portions of bounding faces of polyhedron Q which are located outside of P .

If W is the union of P and Q , then the boundary of W is composed of portions of those faces of P that are in the exterior of Q and those of Q that are in the exterior of P. If W is the intersection of polyhedra P and Q , then the boundary of W is composed of portions of those faces of P which are in the interior of Q and those of Q which are in the interior of P. If W is the subtraction of polyhedra Q from P , then the boundary of W is composed of portions of those faces of P which are in the exterior of Q and those of Q which are in the interior of P.

However, the above relationships hold true only in situations with no degenerate cases. Those cases with degeneracy can first be resolved by the perturbation method discussed in Chapter 7. Therefore, our discussion assumes a degeneracy-free situation.

The relationship between the edges of a set-combined polyhedron and the edges of its source polyhedra can also be generally described by the following statements:

$$\text{If } W = P \cup Q, \text{ then } E(W) = \{E(P) \cap \overline{Q}\} + \{E(Q) \cap \overline{P}\} + \{F(Q) \cap F(P)\}. \tag{9.10}$$

$$\text{If } W = P \cap Q, \text{ then } E(W) = \{E(P) \cap Q\} + \{E(Q) \cap P\} + \{F(Q) \cap F(P)\}. \quad (9.11)$$

$$\text{If } W = P - Q, \text{ then } E(W) = \{E(P) \cap \overline{Q}\} + \{E(Q) \cap P\} + \{F(Q) \cap F(P)\}. \quad (9.12)$$

Here $E(P) \cap Q$ denotes the portions of edges of P that are interior to Q, and $E(P) \cap \overline{Q}$ denotes the portions of edges of P which are in the exterior of Q. We also use $F(Q) \cap F(P)$ to denote the set of edges created by the intersection of a face of P with a face of Q .

If the polyhedron W is the union of P and Q , then the set of edges of W is composed of those portions of edges of P that are exterior to Q and those portions of edges of Q that are exterior to P as well as the edges formed by the intersection of faces of P with faces of Q. If the polyhedron W is the intersection of P and Q , then the set of edges of W is composed of those portions of edges of P that are interior to Q and those portions of edges of Q that are interior to P as well as the edges which formed by the intersection of faces of P with faces of Q . If the polyhedron W is the subtraction of Q from P , then the set of edges of W is composed of those portions of edges of P that are exterior to Q and those portions of edges of Q that are interior to P as well as the edges formed by the intersection of faces of P with faces of Q .

An edge-face-penetration procedure can perform the task of discovering the intersections between edges and faces of two polyhedra. Afterward, the edges of each polyhedra are cut into segments that are classified by types as either $\{E(Q) \cap \overline{P}\}$, $\{E(Q) \cap P\}$, $\{E(P) \cap \overline{Q}\}$, or $\{E(P) \cap Q\}$. A method is described in Chapter 7 for performing face-face intersection between two polyhedra. This operation solves the intersections between faces or between edges and faces of two polyhedra. By using the perturbation method proposed in the same chapter, we can perform a face-face intersection in a degeneracy-free situation. It follows, then, that by using the face-face intersection method and following the principles above, we can quite readily find the integral of a set-combined polyhedron.

As we have explained before, an integral over a polyhedron can be calculated by means of its edges. We can, therefore, calculate the integral of a set-combined polyhedron W by scanning through its edges discovered from its source polyhedra

following the rules described above. For instance, if W is the intersection of P and Q, then the set of edges of W is composed of three classes of segments: (1) $= \{E(P) \cap Q\}$, (2) $= \{E(Q) \cap P\}$, and (3) $= \{F(Q) \cap F(P)\}$. If $E = (v_m, v_m)$ is a segment of class (1), then E is an edge or a portion of an edge of P. The two consecutive faces F_r and F_l associated with E are two boundary faces of P. Edge E defines two triangles, one with respect to the highest indexed vertex v_0^r of F_r and the other with respect to the highest indexed vertex v_0^l of F_l. The two triangles eventually constitute two tetrahedra with respect to the origin and contribute part of an integrated amount to the final integral.

For a segment E of class (2), E can be an edge or part of an edge of Q. Edge E is also associated with two consecutive boundary faces, F_r and F_l. Edge E, with a fixed vertex v_r^o of face F_r and a fixed vertex v_l^o of face F_l, defines two triangles, that eventually constitute two tetrahedra with respect to the origin. Part of an integrated amount can also be accumulated from these two tetrahedra.

For a segment E of class (3), E is the intersection of a face F_P of P with a face F_Q of Q. Edge E is thus associated with the two faces F_P and F_Q. Edge E, with a fixed vertex v_P^o of face F_P and a fixed vertex v_Q^o of face F_Q, constitutes two triangles that also eventually constitute two tetrahedra with respect to the origin. Part of an integrated amount can also be calculated from these two triangles.

If W is the union of P and Q, then the first two classes of edges are different from that of the union of P and Q, while the third class remains the same. If W is the subtraction of Q from P, then the first two classes of edges are different and the third class is the same. However, an integral over the union or the subtraction of two polyhedra can still be calculated using the same principle described above. For a segment $E\text{in}\{E(P) \cap \overline{Q}\}$, edge E is still a portion of an edge of P and is associated with two faces F_r and F_l of P. Part of an integral can thus be calculated separately on the two triangles defined by E with a fixed vertex of face F_r and a fixed vertex of face F_l. The same calculation can also be performed for a segment $E\text{in}\{E(Q) \cap \overline{P}\}$.

Chapter 10

Integral Over a Polyhedron in R^m Space

To recognize the need in mechanized program analysis for an efficient method to compute the volume of a nonconvex polyhedron in m-dimensional space (R^m), we have to consider the problem of determining the probability that a conditional will yield a "true" or a "false" value from a path in a loopless program consisting of assignments and embedded or cascading conditionals. Let us assume that all variables in a program have known upper and lower bounds, and that the boolean expression in a conditional is formed by a conjunction of linear inequalities. When each boolean expression in the path consists of linear inequalities, the probability of taking that path may be determined by the ratio of the volumes of two polyhedra P_1 and P_2, where P_1 is the polyhedron representing the conjunction of linear inequalities along the path and P_2 is the polyhedron representing the bounds of the variables. In this case, being able to compute the volume of an R^m polyhedron is the key to the solution of the problem.

The symbolic method proposed in the previous sections can be extended to help solve the probability analysis problem described above. With appropriate generalization, we can apply this method to the calculation of the integral properties of a nonconvex R^m polyhedron. For instance, Equation (13), previously defined as giving a direct evaluation of an integral over a unit orthogonal tetrahedron, needs to be generalized to evaluate an integral on a unit orthogonal m-simplex, instead. In addition to computing the volume of an R^m polyhedron, we can also use this method to calculate an integral of an arbitrary polynomial function. For example, if the probability density in the above problem is not distributed uniformly through the entire R^m space, we would need to calculate the mass of polyhedra P_1 and P_2 instead of their volume, making the density distribution function equal to the probability density function.

10.1 Evaluating an integral over an m-simplex

We can describe the integral properties of a polyhedron Q in R^m by the expression

$$I = \int_Q f(\mathbf{x})\mathbf{dx} \tag{10.1}$$

where the integrand f is a polynomial function defined in R^m and $\mathbf{dx}$ is short for the differential $dx_1 dx_2 .. dx_m$. A linear transformation $\mathcal{G}$ is defined in R^m that maps a vector $\mathbf{u}$ into a vector $\mathbf{x}$ by $\mathcal{G} : \mathbf{u} = (u_1, u_2, .., u_m) \Rightarrow \mathbf{x} = (x_1, x_2, .., x_m)$ as

$$\begin{cases} x_1 & = g_1(u_1, u_2, .., u_m) \\ x_2 & = g_2(u_1, u_2, .., u_m) \\ .. & = \quad ....., \\ x_m & = g_m(u_1, u_2, .., u_m) \end{cases}$$

This transforms Equation (10.1) into

$$I = \int_Q f(\mathcal{G}(\mathbf{u}))|J|\mathbf{du}, \tag{10.2}$$

where the Jacobian J is the determinant of an $m \times m$ matrix

$$J = \begin{vmatrix} \partial g_1/\partial u_1 & \partial g_1/\partial u_2 & ... & \partial g_1/\partial u_m \\ \partial g_2/\partial u_1 & \partial g_2/\partial u_2 & ... & \partial g_2/\partial u_m \\ & & ... & \\ \partial g_m/\partial u_1 & \partial g_m/\partial u_2 & ... & \partial g_m/\partial u_m \end{vmatrix} \tag{10.3}$$

A polynomial f in R^m can be represented as

$$f(\mathbf{x}) = \sum_{n_1, n_2, .., n_m} x_1^{n_1} x_2^{n_2} .. x_m^{n_m}, \qquad \text{where } n_i\text{'s are integers.}$$

Similarly, to compute the integral in Equation (10.1), we consider only one term

$$I = \int_Q x_1^{n_1} x_2^{n_2} .. x_m^{n_m} \mathbf{dx}. \tag{10.4}$$

A brief introduction to the topology of polyhedra is included in Appendix C. The notation used below is based on the presentation there. Let $s^m = (O, v_1, ..., v_m)$

be an m-simplex with $m+1$ vertices, where O is the origin and the coordinates of the vertices are

$$\begin{cases} O & = & < 0, 0, ..., 0 > \\ v_1 & = & < x1, 1, x1, 2, ..., x1, m > \\ & & ..... \\ v_m & = & < xm, 1, xm, 2, ..., xm, m > \end{cases} \tag{10.5}$$

The definitions of m-dimensional simplices (m-simplex) and complexes (m-complex) can also be found in the Appendix. Let $\mathbf{v_i}$ represent a column vector from the origin to a vertex v_i. An $m \times m$ linear transformation T can thus be defined by the set of vectors $\{\mathbf{v_i}\}$, such that

$$T = (\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_m}) \tag{10.6}$$

Under this transformation, an orthogonal unit m-simplex $W = (v_o', v_1', ..., v_m')$ with coordinates

$$\begin{cases} v_o' & = & (0, 0, ..., 0), \\ v_1' & = & (1, 0, ..., 0), \\ & & ..... \\ v_m' & = & (0, 0, ..., 1), \text{ respectively} \end{cases} \tag{10.7}$$

is transformed into the original m-simplex s^m. Based on the above transformation, the integral in (10.4) becomes

$$\begin{aligned} I &= \int_{s^m} x_1^{n_1} x_2^{n_2} ... x_m^{n_m} \mathbf{dx}, \\ &= \|T\| \int_W (g_1)^{n_1} (g_2)^{n_2} ... (g_m)^{n_m} \mathbf{du}, \end{aligned} \tag{10.8}$$

where the transformation functions g_i's are defined as

$$\begin{aligned} g_1(u_1, ..., u_m) &= (x_1^1 u_1 + x_1^2 u_2 + .. + x_1^m u_m) \\ g_2(u_1, ..., u_m) &= (x_2^1 u_1 + x_2^2 u_2 + .. + x_2^m u_m) \\ &..... \\ g_m(u_1, ..., u_m) &= (x_m^1 u_1 + x_m^2 u_2 + .. + x_m^m u_m) \end{aligned}$$

We generalize the formula presented in the previous chapter to the following formula, which resolves an integral of a polynomial $x_1^{n_1} x_2^{n_2} .. x_m^{n_m}$ over a unit orthogonal

m-simplex as described in Equation (10.7):

$$\int_W x_1^{n_1} x_2^{n_2} ... x_m^{n_3} \mathbf{dx} = \frac{n_1!\ \ n_2!\ \ ...\ \ n_m!}{(n_1 + ... + n_m + m)!} \tag{10.9}$$

An arbitrary m-simplex $(v_o, v_1, .., v_m)$ in which v_o is not the origin can always be transformed into an orthogonal unit m-simplex by means of the same transformation, if only the vector $\mathbf{v}_i$ denotes a vector from v_o to v_i.

$$T = (\mathbf{v}_1, \quad \mathbf{v}_2, \quad ..., \quad \mathbf{v}_m) \tag{10.10}$$

An integral over an m-simplex can then be evaluated symbolically by Equation (10.9).

10.2 Evaluation of an R^m transformation

Next we discuss how to calculate the determinant of an $m \times m$ matrix as defined in Equation (10.10). We present a simple method to accomplish this operation. We first define a set of new bases $(\mathbf{f}_1, \mathbf{f}_2, .., \mathbf{f}_m)$ from the simplex $s^m = (v_o, v_1, ..., v_m)$ in the previous discussion, such that

$$\begin{cases} \mathbf{f}_1 = \dfrac{\mathbf{v}_1}{|\mathbf{v}_1|} \\ \mathbf{f}_2 = \dfrac{\mathbf{v}_2 - (\mathbf{v}_2 \bullet \mathbf{f}_1)\mathbf{f}_1}{|\mathbf{v}_2 - (\mathbf{v}_2 \bullet \mathbf{f}_1)\mathbf{f}_1|} \\ ... = ... \\ \mathbf{f}_m = \dfrac{\mathbf{v}_m - (\sum_{i=1}^{m-1} \mathbf{v}_m \bullet \mathbf{f}_i)\mathbf{f}_i}{|\mathbf{v}_m - (\sum_{i=1}^{m-1} \mathbf{v}_m \bullet \mathbf{f}_i)\mathbf{f}_i|} \end{cases} \tag{10.11}$$

Here "$\bullet$" denotes the inner product of two vectors. Basis $(\mathbf{f}_1, \mathbf{f}_2, .., \mathbf{f}_m)$ is a set of orthonormal bases, that is

$$|\mathbf{f}_i \bullet \mathbf{f}_j| = 1 \text{ if } i = j,$$
$$|\mathbf{f}_i \bullet \mathbf{f}_j| = 0 \text{ if } i \neq j, \text{ for } i, j = 1, .., m$$

We know that the determinant of a matrix is not altered by adding a multiple of one column to another column. Therefore, the determinant of the transformation matrices

$$|T| = |(\mathbf{v}_1 \quad ...\mathbf{v}_i \pm k\mathbf{v}_j \quad ... \quad \mathbf{v}_m)| \tag{10.12}$$

are constant for arbitrary number k. On the bases of Equation (10.12), we can shift the contents of the transformation matrix T in Equation (10.10) to T' in the following statement without altering its determinant:

$$T' = (\mathbf{v_1}, \quad \mathbf{v_2} - (\mathbf{v_2} \bullet \mathbf{f_1})\mathbf{f_1}, \quad \mathbf{v_3} - (\mathbf{v_3} \bullet \mathbf{f_1})\mathbf{f_1} - (\mathbf{v_3} \bullet \mathbf{f_2})\mathbf{f_2}, \quad .... \quad) \tag{10.13}$$

The matrix T' is based on the new bases $(\mathbf{f_1}, \mathbf{f_2}, .., \mathbf{f_m})$

$$T' = (h_1\mathbf{f_1}, \quad h_2\mathbf{f_2}, \quad ... \quad , h_m\mathbf{f_m}) \tag{10.14}$$

where each scalar h_i is the length of the i_{th} column vector, defined as

$$h_i = |\mathbf{v}_i - \sum_{j=1}^{j-1} (\mathbf{v}_i \bullet \mathbf{f}_j)\mathbf{f}_j|$$

Since $(\mathbf{f_1}, \mathbf{f_2}, .., \mathbf{f_m})$ is a set of orthonormal bases, the determinant of matrix $F = (\mathbf{f_1}, \mathbf{f_2}, .., \mathbf{f_m})$ is either ± 1. Therefore, the determinant of matrix T' is

$$|T'| = \pm h_1 h_2 ... h_m$$

and, therefore, the determinant of matrix T is

$$|T| = \pm h_1 h_2 ... h_m$$

We interpret the physical meaning of the scalars $h_1, h_2, ..., h_m$ in the following discussion. We first recall that $(\mathbf{f_1}, \mathbf{f_2}, .., \mathbf{f_m})$ are defined by extracting from a vector $\mathbf{v}_i$ the component that is orthogonal to the previously defined bases $\mathbf{f_1}, .., \mathbf{f}_{i_1}$, such that

$$\begin{aligned} \mathbf{f_1} &= \frac{\mathbf{v_1}}{|\mathbf{v_1}|} \\ \mathbf{f_2} &= \frac{\mathbf{v_2} - (\mathbf{v_2} \bullet \mathbf{f_1})\mathbf{f_1}}{h_2} \\ ... &= ..... \\ \mathbf{f_m} &= \frac{\mathbf{v_m} - (\sum_i \mathbf{v_m} \bullet \mathbf{f_i})\mathbf{f_i}}{h_m} \end{aligned} \tag{10.15}$$

We can interpret vector $\mathbf{v_1}$ as a 1-simplex $s^1 = (v_o, v_1)$ consisting of two vertices. The scalar h_1 is the length of the vector $\mathbf{v_1}$, which is also called generally the 'volume' of

simplex s^1 and is expressed as $|s^1| = |\mathbf{v}_1|$. Vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ expand a "2-simplex" consisting of three vertices $s^2 = (v_o, v_1, v_2)$; this arrangement forms a '2-cone' built on a 'base' $s^1 = (v_o, v_1)$ with vertex v_2. In Appendix C we can find the definitions of "cone" and "base." We know that the length

$$h_2 = |\mathbf{v}_2 - (\mathbf{v}_2 \bullet \mathbf{f}_1)\mathbf{f}_1|$$

marks the orthogonal distance from the vertex v_2 to the base (v_o, v_1), i.e., the distance between v_2 and the line containing (v_o, v_1). The "volume" (the area) of simplex s^2 is

$$|s^2| = \frac{1}{2} h_2 |s^1|$$

Vectors $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ expand a 3-simplex consisting of four vertices $s^3 = (v_o, v_1, v_2, v_3)$, forming a 3-cone on $s^2 = (v_o, v_1, v_2)$ with vertex v_3. As we can easily see, the length

$$h_3 = |\mathbf{v}_3 - (\mathbf{v}_3 \bullet \mathbf{f}_1)\mathbf{f}_1 - (\mathbf{v}_3 \bullet \mathbf{f}_2)\mathbf{f}_2|$$

equals the distance from v_3 to the plane containing the base $s^2 = (v_o, v_1, v_2)$, and the volume of simplex s^3 is thus

$$|s^3| = \frac{1}{3} h_3 |s^2|$$

Finally, a set of m vectors $\mathbf{v}_1, \mathbf{v}_2$, ... and $\mathbf{v}_m$ expands an m-simplex s^m consists of $m+1$ vertices to form an m-cone built on its previous simplex, $s^{m-1} = (v_o, v_1, ..., v_{m-1})$ with vertex v_m. The volume of s^m is therefore equal to

$$\frac{1}{m} h_m |s^{m-1}|,$$

where $|s^{m-1}|$ is the volume of its base and h_m is the distance from v_m to its base.

10.3 Integration over an R^m polyhedron

An R^m polyhedron Q is an m-complex whose boundary is a set of $(m-1)$-complexes denoted by $\{B_i^{m-1}\}$. The boundary of a $(m-1)$-complex $B_{i_1}^{m-1}$ is again a set of $(m-2)$-complexes denoted by $\{B_{i_1,i_2}^{m-2}\}$, etc. Let Q be represented as $Q =$

$\{B_i^{m-1}\}$ where $\{B_i^{m-1}\}$ are the boundaries of Q. Let p_m be a point in R^m which forms an m-cone C_i^m with respect to each boundary B_i^{m-1} of Q. Point p_m, therefore, forms a set of m-cones $\{C_i^m = p_m B_i^{m-1}\}$ with respect to all the boundaries of Q. An integral over Q thus equals the sum of appropriately signed integrals over the set of cones $\{C_i^m\}$, such that

$$\int_Q f(x_1, .., x_n)\mathbf{dx} = \sum_{C_i^m} S(C_i^m) \int_{C_i^m} f(x_1, .., x_n)\mathbf{dx}$$

where the sign function $S(C_i^m)$ is determined by the side of B_i^{m-1} on which point p_m is located.

An R^m polyhedron is an m-complex. The boundaries of an R^m polyhedron are a set of $(m-1)$-complexes. Similarly, the boundaries of a boundary of an R^m polyhedron is a set of $(m-2)$-complexes. The subspace of R^m in which an $(m-1)$-complex is embedded is a super plane which can be described by a superplane equation $\Gamma(\mathbf{x}) = 0$, where

$$\gamma_m x_m + \gamma_{m-1} x_{m-1} + ... + \gamma_1 x_1 + \gamma_0 = 0$$

and $< \gamma_m, \gamma_{m-1}, ..., \gamma_1 >$ is the unit outward normal vector of the super plane.

Let $\Gamma_i(\mathbf{x}) = 0$ be a superplane with an embedded B_i^{m-1} boundary. The distance from a point $p_m =< x_1^m, x_2^m, .., x_m^m >$ to a superplane is the absolute value of the result of substituting the coordinates of the point into the plane equation of the superplane. A 'signed' distance from a point $p_m =< x_1^m, x_2^m, .., x_m^m >$ to a boundary B_i^{m-1} is a direct result of substituting the coordinates of p_m into the plane equation, that is

$$h_m = \Gamma_i(p_m) = \gamma_m^i x_m^m + \gamma_{m-1}^i x_{m-1}^{m-1} + ... + \gamma_1^i x_1^1 + \gamma_0^i$$

The sign function $S(C_i^m)$ is defined as the sign of the signed distance h_m, that is

$$S(C_i^m) = \text{Sign}(h_m) = \text{Sign}(\Gamma_1(p_m))$$

If h_m has positive sign, then point p_m is located on the positive side of B_i^{m-1}; if h_m has negative sign, then p_m is on the negative side.

Let B^{m-1} denote a boundary of Q, and $\{B_i^{m-2}\}$ denote the set of boundaries of B^{m-1}. We can dissect the boundary B^{m-1}, which is an $(m-1)$-complex, into a set of $(m-1)$-cones $\{C_i^{m-1} = p_{m-1}B_i^{m-2}\}$ by selecting a point p_{m-1} out of the superplane in which B^{m-1} is embedded. Each cone in $\{C_i^{m-1} = p_{m-1}B_i^{m-2}\}$ is constructed on the bases of $\{B_i^{m-2}\}$ with point p_{m-1}. The volume of B^{m-1} equals the summation of the appropriately signed volume of each cone, that is

$$\sum_i \frac{1}{m-1} h_{m-1} |B_i^{m-2}|$$

where $|B_i^{m-2}|$ is the volume of a base B_i^{m-2} and h_{m-1}'s are signed distances from p_{m-1} to the base. Here 'distance' is considered in the subspace in which B^{m-1} is embedded.

Let B_i^{m-1} represent the i_{th} boundary of Q and $B_{i,j}^{m-2}$ represents the j_{th} boundary of B_i^{m-1}. The subspace of R^m in which a $(m-2)$-complex $B_{i,j}^{m-2}$ is embedded can be described by two superplanes that lie orthogonal to each other

$$\Gamma_i(\mathbf{x}) = \gamma_m^i x_m + \gamma_{m-1}^i x_{m-1} + ... + \gamma_1^i x_1 + \gamma_0^i = 0, \tag{10.16}$$

and

$$\Gamma_{i,j}(\mathbf{x}) = \gamma_m^j x_m + \gamma_{m-1}^j x_{m-1} + ... + \gamma_1^j x_1 + \gamma_0^j = 0, \tag{10.17}$$

$\Gamma_i(\mathbf{x}) = 0$ is the superplane containing B_i^{m-1} and $\Gamma_{i,j}(\mathbf{x}) = 0$ is an orthogonal superplane so that the intersection of the two is the subspace containing $B_{i,j}^{m-2}$. Here 'orthogonal' means that the inner product of the normal vectors of the two faces equals zero, that is

$$(\gamma_m^i, \gamma_{m-1}^i, ..., \gamma_1^i) \bullet (\gamma_m^j, \gamma_{m-1}^j, ..., \gamma_1^j) = 0$$

Since the two planes in (10.16) and (10.17) are orthogonal, the distance h_{m-1} from p_{m-1} to the base $\{B_{i,j}^{m-2}\}$ equals

$$h_{m-1} = \Gamma_{i,j}(p_{m-1}) = \gamma_m^j x_m^{m-1} + \gamma_{m-1}^j x_{m-1}^{m-1} + ... + \gamma_1^j x_1^{m-1} + \gamma_0^j.$$

Generally speaking, the subspace in which a $(m-j)$-complex is embedded can be described by a set of j superplanes as the followings:

$$\begin{cases} \Gamma_1(\mathbf{x}) = 0 \\ \Gamma_2(\mathbf{x}) = 0 \\ \quad .... \\ \Gamma_j(\mathbf{x}) = 0 \end{cases} \tag{10.18}$$

The subspace in which the boundary of a $(m-j)$-complex is embedded can be described by adding one more orthogonal superplane to the original set of planes, and resulting in

$$\begin{cases} \Gamma_1(\mathbf{x}) = 0 \\ \Gamma_2(\mathbf{x}) = 0 \\ \quad .... \\ \Gamma_j(\mathbf{x}) = 0 \\ \Gamma_{j+1}(\mathbf{x}) = 0 \end{cases} \tag{10.19}$$

The signed distance h_{m-j} from a point $p_{m-j} =< x_1^{m-j}, ..., x_m^{m-j} >$ of B^{m-j} to one of its boundaries $\{B_i^{m-j-1}\}$ equals $\Gamma_{j+1}(p_{m-j})$, that is

$$h_{m-j} = \Gamma_{j+1}(p_{m-j})$$

Let p_m be a point in R^m . Let $\{B_{i_1}^{m-1}\}$ be the boundaries of Q and $\{p_{m-1}^{i_1}\}$ be the local points selected respectively out of the subspaces where $\{B_{i_1}^{m-1}\}$ are embedded. Let $\{B_{i_1,i_2}^{m-1}\}$ be the boundaries of $B_{i_1}^{m-1}$ and $\{p_{m-2}^{i_1,i_2}\}$ be the points selected respectively from the subspaces where $\{B_{i_1,i_2}^{m-1}\}$ are embedded, and so on. The entire procedure described above can be used to dissect a polyhedron Q into a set of m-simplices $\{(p_o, p_1^{i1}, p_2^{i1,i2}, ..., p_m^{i1,..,im})\}$.

Since a volume integral over a unit orthogonal m-simplex can be evaluated directly through Equation (10.9), it follows that the determinant of an $m \times m$ matrix can also be readily calculated. The volume integral over an m-simplex $s = (p_o, p_1^{i1}, p_2^{i1,i2}, ..., p_m^{i1,..,im})$ can then be evaluated directly, by means of a simplex-dependent transformation $T(s)$ defined as in Equation (10.6). A volume integral over a polyhedron

Q thus equals the sum of the 'appropriately signed' integrals over all the simplices, where the sign for each simplex is contained in the signed distances from a local highest indexed vertex to its base.

Chapter 11

Conclusion

The goal of this research has been to explore powerful computational means to improve and perfect current solid modeling systems so that engineers can rely on them to easily and efficiently generate complete designs, perform automatic pre-checks, and calculate engineering-related properties.

Toward the realization of this goal, we present two basic theorems, a transition criterion and a singularity criterion, that support the theoretical foundations of solid modeling and permit verification of computational techniques. A new solid representation scheme is also proposed. Taking a new view of computerized geometries, it allows geometrical set operations to be decomposed into local suboperations. An algorithm for performing face-face intersection in a set operation is presented as a new way of locating the intersections of two polyhedra without edge-face penetration detections as conventionally required. This algorithm drastically reduces the computational burden on a set operator in a solid modeling system. Finally, a symbolic method is presented for efficiently calculating certain important engineering-related properties. The dissection technique at the heart of this method makes the overall implementation remarkably easy.

This ease of implementation is the final aim of each of the innovations and techniques listed above, not just in the sense of mathematical clarity or efficiency, but most important, in the sense of practical utility. Point-polygon- and point-polyhedron-enclosure detections, for example, are two very time consuming operations which occur very frequently in hidden-line elimination and object interference detection. An efficient execution of these two operations can significantly improve both the speed of object interference checking and the performance of the user interface in a solid modeling system. The transition criterion proposed in Chapter 2 leads to a direct and efficient method for solving two-dimensional point-enclosure problems. The singularity

criterion proposed in Chapter 4 leads to a theoretically exact solution to all the singularities possibly encountered in a three-dimensional point-enclosure problems.

We also categorize all singularities possibly encountered in 3D point enclosure detection, so that object interference detection in a solid modeling system can be implemented in a clean and exact way. And we introduce an efficient algorithm for 3D point-enclosure detection which reduces the number of 2D point-enclosure detections required during the computation.

Similarly, set operations on solids normally require several time-consuming local operations, such as edge-face-penetration, point-polygon enclosure, point-polyhedron enclosure, edge-edge intersection, polygon-polygon intersection, face-face intersection, and polyhedron-polyhedron intersection. Good techniques for these operations can improve the performance of a solid modeling system. Among these techniques, perturbation is a method which can be used to transform two objects into a singularity-free situation so that edge-edge intersection and face-face intersection can be performed easily. A set of rules is generated, as a result of perturbation, that can be followed in implementing a set operation in a solid modeling system. This set of rules makes the implementation of edge-edge intersection and face-face intersection executors in a solid modeling system short and clean.

A scheme called skeletal polyhedron representation is likewise proposed in this thesis, as a new means of representing solids. Skeletal polyhedron representation describes solids based on the relationship between their vertices rather than by the conventional scheme of boundary representation. This scheme allows set operations on solids to be performed in a more efficient and convenient way. A detailed discussion of 2D and 3D set operations on solids on the basis of skeletal representation is presented in Chapters 5-7.

Finally, our work seeks to fulfill the specification that a useful solid modeling system should supply not only efficient means of handling objects, but also efficient techniques for performing engineering-related calculations. To do this, we presented a symbolic method for calculating the integral properties of an arbitrary nonconvex

polyhedron in Chapter 8. This method not only is efficient but also gives an exact result of an integral of an arbitrary polynomial function over a polyhedron. Besides, the implementation is very simple because of the systematization of the method.

By taking advantage of the features of the above symbolic method, we present in Chapter 9 a similar method for directly calculating the integral properties of a set-combined polyhedron, that is a polyhedron which is the union, intersection, or subtraction of two polyhedra. This method avoids the necessity of rebuilding the resultant polyhedron through a set operator. The symbolic method in Chapter 8 is not confined by the dimensionality of the space, it can be generalized to permit the calculation of the integral properties of a polyhedron in m-dimensional space.

This decade is proving to be an era of intense interest in computer-aided geometric design, an activity attracting people from many sectors including computing and manufacturing. As the discipline more firmly establishes valid principles and standard techniques, it will undoubtedly come to dominate computer-aided manufacturing as well as a wide range of other applications. To provide increased capability for future applications, we must continue making advances in such areas as mathematical representation, logical structuring, management of geometric data, development of appropriate geometry standards, and improvement in user interfaces through interactive graphics techniques.

References

[1] Appel, A., "Some Techniques for Shading Machine Renderings of Solids," SJCC 37-45, 1968.

[2] Ballard, D.H., "Strip Trees: A Hierarchical Representation for Curves," Communications of the ACM, Vol. 24. No. 5, May 1981.

[3] Barnhill, R.E. and R.F. Riesenfeld, "Computer-Aided Geometric Design," Eds. Academic Press, 1974.

[4] Baumgart, B.G., "Geometric Modeling for Computer Vision," Stanford Artificial Intelligence Laboratory Memo Aim 249, October 1974.

[5] Baumgart, B.G., "Winged Edge Polyhedron Representation," Stanford Artificial Intelligence Laboratory Memo Aim 179, October 1972.

[6] Bentley, J.L. and W. Carruthers, "Algorithms for Testing the Inclusion of Points in Polygons," 18th Annual Allerton Conf. on Communication, Control and Computing, ACM, 1980.

[7] Boyse, J.W., "Interference Detection Among Solids and Surfaces," Communications of the ACM, Vol. 22, No. 1, 1979.

[8] Cohen, J. and T. Hickey, "Two Algorithms for Determining Volumes of Convex Polyhedra," J. ACM, Vol. 26, No. 3, July 1982.

[9] Eastman, C., and Y. Kalay, "An Algorithm for Spatial Set Manipulations of Solid Objects," Research Report, Dept. of Archi, Carnegie-Mellon Univ., July 1980.

[10] Eastman, C., J. Lividini, and D. Stroker, "A Database for Designing Large Physical Systems," Nat. Compu. Conf. Proc., 1975.

[11] Edmonds, J., "A Combinatorial Representation for Polyhedral Surfaces," Notices Amer. Math. Soc. Vol. 7, 646, 1960.

[12] Franklin, W.R., "Efficient Polyhedron Intersection and Union," Graphics Interface,

73-80, 1982.

[13] Goldstein, E. and R. Nagle, "3D Visual Simulation," Simulation 16, 25-31, January. 1971.

[14] Hanarahan, P.M., "Creating Volume Models from Edge-Vertex Graphs," ACM Computer Graphics Vol. 16, No. 3, 1982.

[15] Jackins, C. L. and S.L. Tanimoto, "Oct-Trees and Their Use in Representing Three-Dimensional Objects," Computer Graphics and Image Processing, 14, 3, 249-270, November 1980.

[16] Kalay, Y.E., "Determining the Spatial Containment of a Point in General Polyhedra," Computer Graphics and Image Processing 19, 303-334, 1982.

[17] Lee, Y.T. and A.A.G. Requicha, "Algorithms for Computing the Volume and Other Integral Properties of Solids, I. Known Methods and Open Issues," Communications of the ACM, Vol. 25, No. 9, September 1982.

[18] Lee, Y.T. and A.A.G. Requicha, "Algorithms for Computing the Volume and Other Integral Properties of Solids, II. A Family of Algorithms Based on Representation Conversion and Celluar Approximation," Communications of the ACM, Vol. 25, No. 9, September 1982.

[19] Lien, S.L. and J. Kajiya, "A Transition Criterion for Vertex to Surface Comparison," ACM Transactions on Graphics, (submitted), 1983.

[20] Lien, S.L. and J. Kajiya, "Resolving Singularities in Point-Polyhedra Enclosure Detection," ACM Transactions on Graphics, (submitted), 1984.

[21] Lien, S.L. and J. Kajiya, "A Symbolic Method for Calculating the Integral Properties of Arbitrary Nonconvex Polyhedra," IEEE Computer Graphics and Applications, October 1984.

[22] Maruyama, K., "A Procedure to Determine Intersections Between Polyhedra Objects," Inter. J. of Compu. and Infor. Sci., Vol.1, No. 3, 1972.

[23] Putnam, L.K. and P.A. Subrahmanyam, "Computation of the Union, Intersection

and Difference of N-dimensional Objects via Boundary Classification," Department of Computer Science, Univ. of Utah, 1981.

[24] Requicha, A.A.G. and H.B. Voelcker, "Solid Modeling: A Historical Summary and Contemporary Assessment," IEEE Computer Graphics and Applications, Vol. 2, No. 2, March 1982.

[25] Requicha, A.A.G. and H.B. Voelcker, "Solid Modeling: Current Status and Research Directions," IEEE Computer Graphics and Applications, Vol. 3, No. 7, Oct. 1983.

[26] Requicha, A.A.G. and R.B. Tilove, "Mathematical Foundations of Constructive Solid Geometry: General Topology of Closed Regular Sets," Production Automation Project, TM-27a, University of Rochester, 1978.

[27] Roth, S.D., "Ray Casting for Modeling Solids," Computer Graphics and Image Processing 18, 109-144, 1982.

[28] Sutherland, I.E., R.A. Sproull, and R.A. Schumacker, "A Characterization of Ten Hidden-Surface Algorithms," Computing Surveys, Vol. 6, No. 1, 1974.

[29] Voelcker, H.B. and A.A.G. Requicha, "Geometric Modeling of Mechanical Parts and Processes," Computer 10, December 1977.

[30] Wesley, M. A., "Construction and Use of Geometric Models," Computer Aided Design, Lecture Notes in Computer Science 89, 79-136, 1980.

[31] Whitted, J.T., "A Scan Line Algorithm for Computer Displaying of Curved Surfaces," Computer Graphics (SIGGRAPH 78 Supplement) Vol. 13(3), August 1978.

[32] Wilkinson, J.H., "Rounding Errors in Algebraic Processes," Prentice-Hall, INC., 1963.

Appendix A: Derivation of the integral formula

The Beta function and Gamma function are defined as:

$$B(n_1+1, n_2+1) = \int_0^1 x^{n_1}(1-x)^{n_2}dx$$
$$\Gamma(n+1) = \int_0^\infty e^{-x}x^n dx = n!$$
$$B(m,n) = \frac{\Gamma(m)\Gamma(n)}{\Gamma(m+n)} = \frac{(m-1)!(n-1)!}{(m+n-1)!}$$

The derivation of the formula for evaluating the integral of a polynomial $x^{n_1}y^{n_2}z^{n_3}$ over an orthogonal unit tetrahedron proceeds as follows:

$$\begin{aligned}
&\int_W x^{n_1}y^{n_2}z^{n_3}dxdydz \\
=&\int_0^1\int_0^{1-z}\int_0^{1-z-y} x^{n_1}y^{n_2}z^{n_3}dxdydz \\
=&\frac{1}{n_1+1}\int_0^1\int_0^{1-z}(1-z-y)^{n_1+1}y^{n_2}z^{n_3}dydz \\
=&\frac{1}{n_1+1}\int_0^1\int_0^{1-z}(1-z)^{n_1+1}(1-\frac{y}{1-z})^{n_1+1}y^{n_2}z^{n_3}dydz \\
&\qquad \text{let } Y=\frac{y}{1-z}, \quad y=(1-z)Y, \quad dy=(1-z)dY \\
=&\frac{1}{n_1+1}\int_0^1\int_0^1(1-z)^{n_1+1}(1-Y)^{n_1+1}(1-z)^{n_2}Y^{n_2}z^{n_3}(1-z)dYdz \\
=&\frac{1}{n_1+1}\int_0^1\int_0^1(1-Y)^{n_1+1}Y^{n_2}(1-z)^{n_1+n_2+2}z^{n_3}dYdz \\
=&\frac{1}{n_1+1}\int_0^1 B(n_2+1,n_1+2)(1-z)^{n_1+n_2+2}z^{n_3}dz \\
=&\frac{B(n_2+1,n_1+2)}{n_1+1}\int_0^1(1-z)^{n_1+n_2+2}z^{n_3}dz \\
=&\frac{B(n_2+1,n_1+2)}{n_1+1}B(n_3+1,n_1+n_2+3) \\
=&\frac{1}{n_1+1}\frac{n_2!(n_1+1)!}{(n_1+n_2+2)!}\frac{n_3!(n_1+n_2+2)!}{(n_1+n_2+n_3+3)!} \\
=&\frac{n_1!\quad n_2!\quad n_3!}{(n_1+n_2+n_3+3)!}
\end{aligned}$$

Appendix B: Integral Algorithm

The following is an algorithm for calculating volume, center of mass, and moments of inertia of a polyhedron.

```
procedure Integral ( Q: polyhedron );
var j, k : integer;
    p1, p2, p3 : point;
    E : edge;
    fr, fl:integer;  { right and left consecutive faces  }
    xo, yo, zo , determ, volume: real;
    bary_xo, bary_yo, bary_zo : real; { barycenter  }
    Ixx, Iyy, Izz, Ixy, Iyz, Izx : real; { moment of inertia  }
            { calculating partial bary-center  }
     procedure bary_center( determ: real; p1, p2, p3: point);
     begin    xo:=xo+determ*(p1.x+p2.x+p3.x);
              yo:=yo+determ*(p1.y+p2.y+p3.y);
              zo:=zo+determ*(p1.z+p2.z+p3.z);
     end;
            { calculating partial moment of inertia  }
     procedure moment_of_inertia( determ: real; p1, p2, p3: point);
     begin    Ixx:=Ixx+determ*(p1.x*p1.x+p2.x*p2.x+p3.x*p3.x+
                               p1.x*p2.x+p2.x*p3.x+p3.x*p1.x);
              Iyy:=Iyy+determ*(p1.y*p1.y+p2.y*p2.y+p3.y*p3.y+
                               p1.y*p2.y+p2.y*p3.y+p3.y*p1.y);
```

```
            Izz:=Izz+determ*(p1.z*p1.z+p2.z*p2.z+p3.z*p3.z+
                             p1.z*p2.z+p2.z*p3.z+p3.z*p1.z);
            Ixy:=Ixy+determ*(2*(p1.x*p1.y+p2.x*p2.y+p3.x*p3.y)+
                             p1.x*p2.y+p2.x*p3.y+p3.x*p1.y+
                             p1.y*p2.x+p2.y*p3.x+p3.y*p1.x);
            Iyz:=Iyz+determ*(2*(p1.y*p1.z+p2.y*p2.z+p3.y*p3.z)+
                             p1.y*p2.z+p2.y*p3.z+p3.y*p1.z+
                             p1.z*p2.y+p2.z*p3.y+p3.z*p1.y);
            Izx:=Izx+determ*(2*(p1.z*p1.x+p2.z*p2.x+p3.z*p3.x)+
                             p1.z*p2.x+p2.z*p3.x+p3.z*p1.x+
                             p1.x*p2.z+p2.x*p3.z+p3.x*p1.z);
        end;
    begin { initialization }
        xo:=0; yo:=0; zo:=0; volume:=0;
        Ixx:=0; Iyy:=0; Izz:=0; Ixy:=0; Iyz:=0; Izx:=0;
         for j:=1 to Q.total_edges do
         begin E:=Q.edge[j];
            fr:=E.right_face;
                { p1 is assigned the fixed vertex of face fr }
            p1:=Q.vertex[ Q.edge[ Q.face[fr].eo ].v1 ].p;
                { p2 is assigned the head of edge E }
            p2:=Q.vertex[E.v1].p;
                { p3 is assigned the tail of edge E }
            p3:=Q.vertex[E.v2].p;
                { a transformation is defined on p1, p2, p3, and the origin }
            determ:=determinant(p1, p2, p3);
                { accumulating results from each tetrahedron }
            volume:=volume+determ;
            bary_center(determ, p1, p2, p3);
            moment_of_inertia(determ, p1, p2, p3);
```

```
            fl:=E·left_face;
            p1:=Q·vertex[ Q·edge[ Q·face[fl]·eo ]·v1 ]·p;
                  { notice the exchange of v1 and v2  }
            p2:=Q·vertex[E·v2]·p;
            p3:=Q·vertex[E·v1]·p;
            determ:=determinant(p1, p2, p3);
            volume:=volume+determ;
            bary_center(determ, p1, p2, p3);
            moment_of_inertia(determ, p1, p2, p3);
        end;  { of for j:=1 to Q·total_edges do  }
      volume:=volume/6;
      bary_xo:=xo/24/volume;
      bary_yo:=yo/24/volume;
      bary_zo:=zo/24/volume;
      Ixx:=Ixx/60/volume;
      Iyy:=Iyy/60/volume;
      Izz:=Izz/60/volume;
      Ixy:=Ixy/120/volume;
      Iyz:=Iyz/120/volume;
      Izx:=Izx/120/volume;
end;



procedure Integral ( Q1, Q2: polyhedron; operation:integer );
var j, k : integer;
```

```
    pr, pl, p2, p3 : point;
    E : edge;
    fr, fl:integer;  { right and left consecutive faces  }
    xo, yo, zo , determ, volume: real;
    bary_xo, bary_yo, bary_zo : real; { barycenter  }
    Ixx, Iyy, Izz, Ixy, Iyz, Izx : real; { moment of inertia  }
            { calculating partial bary-center  }
begin { initialization  }
      xo:=0; yo:=0; zo:=0; volume:=0;
      Ixx:=0; Iyy:=0; Izz:=0; Ixy:=0; Iyz:=0; Izx:=0;
       for i:=1 to Q1·total_faces  do
       for j:=1 to Q2·total_faces  do
       face_to_face_cross_examination(i,j);

       for j:=1 to Q·total_edges  do
       begin E:=Q·edge[j];
            fr:=E·right_face;
            fl:=E·left_face;
                { pr is assigned the fixed vertex of face fr  }
            pr:=Q·vertex[ Q·edge[ Q·face[fr]·eo ]·v1 ]·p;
                { pl is assigned the fixed vertex of face fr  }
            pl:=Q·vertex[ Q·edge[ Q·face[fr]·eo ]·v1 ]·p;

             classify(E, p2, p3);
            determ:=determinant(pr, p2, p3);
            volume:=volume+determ;
            bary_center(determ, pr, p2, p3);
            moment_of_inertia(determ, pr, p2, p3);
            determ:=determinant(pl, p3, p2);
            volume:=volume+determ;
```

```
            bary_center(determ, p1, p3, p2);
            moment_of_inertia(determ, p1, p3, p2);
        end;

    volume:=volume/6;
    bary_xo:=xo/24/volume;
    bary_yo:=yo/24/volume;
    bary_zo:=zo/24/volume;
    Ixx:=Ixx/60/volume;
    Iyy:=Iyy/60/volume;
    Izz:=Izz/60/volume;
    Ixy:=Ixy/120/volume;
    Iyz:=Iyz/120/volume;
    Izx:=Izx/120/volume;
end;
```

Appendix C: Introduction to the Topology of Polyhedra

In this section we include a brief introduction to the topology of polyhedra to give the reader some idea of what a *simplex* and a *complex* are. We also introduce notation which is used in this thesis in Chapters 5 and 10.

1 Rectilinear simplexes

Elementary combinatorial topology is concerned with those topological spaces which admit dissection into suitably regular pieces. If we put this the other way around, we need to formulate the concept of a standard space or type of space in a precise manner and then to define the concept of building a space from such pieces or "bricks."

The bricks are called *simplexes*; a simplex is a generalization of an interval (1-simplex), a triangle (2-simplex) or a tetrahedron (3-simplex). Here we are only concerned with subsets of a Euclidean space R^m; the points of R^m can be determined by real m-dimensional position vectors with respect to some chosen Cartesian coordinate system in R^m (we shall use the same notation for a point of R^m and for the vector associated with it). Thus, if $\mathbf{a}$, $\mathbf{b}$ are points of R^m and α, β are real numbers, we may speak of the point $\alpha\mathbf{a} + \beta\mathbf{b}$. In such a context the symbol 0 represents the origin.

We then introduce the preliminary notation necessary for the definition of a p-simplex. A set of (p+1) points of R^m , $\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p$, is said to be a set of *independent* points if the equations

$$\sum_{i=0}^{p} \lambda_i \mathbf{a}^i = 0, \sum_{i=0}^{p} \lambda_i = 0, \tag{1}$$

together imply $\lambda_0 = \lambda_1 = ... = \lambda_p = 0$. This definition clearly describes a property of the points $\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p$themselves and is independent of the choice of origin. A point

$\mathbf{b}$ is said to be *dependent* on the set $\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p$ if there exist $\lambda_0, ..., \lambda_p$ such that

$$\sum_{i=0}^{p} \lambda_i \mathbf{a}^i = \mathbf{b} \text{ and } \sum_{i=0}^{p} \lambda_i = 1. \tag{2}$$

It is then obvious that the set $\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p, \mathbf{b}$ is not independent, so that no point of an independent set is dependent on the others.

If $\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p$ is an independent set and $\mathbf{b}$ is dependent on it, the equation (2) is unique. If $\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p$ is a dependent set and $\mathbf{b}$ is dependent on it, the equation (2) is not unique; moreover, if in this case $\mathbf{b}$ admits an equation (2) with all $\lambda_i > 0$, it admits more than one. First let the set be independent and suppose

$$\mathbf{b} = \sum_{i=0}^{p} \lambda_i \mathbf{a}^i = \sum_{i=0}^{p} \mu_i \mathbf{a}^i \text{ with } \sum_{i=0}^{p} \lambda_i = \sum_{i=0}^{p} \mu_i = 1 \tag{3}$$

Then

$$\sum_{i=0}^{p} (\lambda_i - \mu_i)\mathbf{a}^i = 0 \sum_{i=0}^{p} (\lambda_i - \mu_i) = 0 \tag{4}$$

whence, by equation (1), $\lambda_i = \mu_i$ for $i = 1, ..., p$, and the expression (2) is unique.

Conversely, suppose that the set is not independent. Then there exists numbers ν_i, $i = 1, ..., p$, not all zero, such that $\sum_{i=0}^{p} \nu_i \mathbf{a}^i = 0$, $\sum_{i=0}^{p} \nu_i = 0$. Then if $\mathbf{b} = \sum_{i=0}^{p} \lambda_i \mathbf{a}^i$ we also have

$$\mathbf{b} = \sum_{i=0}^{p} (\lambda_i + \nu_i)\mathbf{a}^i = 0 \text{ and } \sum_{i=0}^{p} (\lambda_i + \nu_i) = 1, \tag{5}$$

so that the expression for $\mathbf{b}$ is not unique.

The set of points dependent on $\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p$ form a Euclidean subspace of R^m . If the set is independent the subspace is of dimension p. We can, therefore, attach to each point of the subspace a unique set of *coordinates*; namely, we attach to $\mathbf{b}$ the coordinates $(\lambda_0, \lambda_1, ..., \lambda_p)$, where expression (APNb) holds. The coordinates $(\lambda_0, \lambda_1, ..., \lambda_p)$ are called the *barycentric* coordinates of $\mathbf{b}$ with respect to the independent set $\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p$. The name "barycentric" is derived from the fact that $\mathbf{b}$ is the center of mass of masses λ_i placed at the points $\mathbf{a}^i$, $i = 0, ..., p$.

The *rectilinear p-simplex*, s^p, with vertices $\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p$ is the set of points dependent on $\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p$ whose barycentric coordinates satisfy $\lambda_i > 0, i = 0, ..., p$. The simplex s^p with vertices $\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p$ may be written $(\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p)$; s^p is said to be *spanned* by its vertices. It is clear that s^p is an open subset in the subspace of R^m consisting of points dependent on $\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p$. For brevity we may call the latter space the subspace determined by $\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p$ and so we write $R(\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p)$.

The closure of s^p, $\mathring{s}^p$, is a closed rectilinear p-simplex and consisting of points of $R(\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p)$ whose barycentric coordinates satisfy $\lambda_i \geq 0, i = 0, .., p$. The frontier of s^p is written $\mathfrak{s}^p$ and is defined as $\mathring{s}^p$-s^p. Then $\mathfrak{s}^p$ consists of those points of $\mathring{s}^p$ for which $\lambda_i = 0$ for some i, $0 \leq i \leq p$. The simplex s^p is said to have *dimension* p; this agrees with its dimension as a Euclidean space. A p-simplex is a simplex of dimension p.

For clearly every point of $\mathring{s}^p$, except the vertices, is the midpoint of a segment lying in $\mathring{s}^p$; this provides a characterization of the vertices. If s^p=$(\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p)$, any subset of $\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p$ is also independent and a simplex spanned by a subset of the vertices is called a *face* of s^p. In particular, s^p is a face of itself; the other faces are called *proper* faces of s^p and have dimension less than p. We write $s^q \prec s^p$ if s_q is a face of s^p.

Each point of $\mathfrak{s}^p$ belongs to precisely one proper face f s^p. Conversely, the points of all proper faces of $s^p$are all points of $\mathfrak{s}^p$. Then $\sum_i^p \lambda_i \mathbf{a}^i = 0$ is a point of $\mathfrak{s}^p$if and only if some λ_i is zero.

2 Simplicial complexes

In this section we describe how simplexes may be fitted together to form more interesting configurations. In this way we shall be able to bring combinatorial methods to bear on spaces of widely differing topological types. The configurations discussed are called *finite geometric simplicial complexes*, abbreviated where there can be no ambiguity to *geometric complex* or just *complex*.

A finite geometric simplicial complex in R^m space is a finite collection, K, of simplexes $\{s_i^p\}$ of R^m , subject to the conditions

$K1$:If $s^p \in K$ and $s^q \prec s^p$, then $s^q \in K$.
$K2$:Distinct simplexes of K are disjoint.

The dimension of K is the maximum of the dimension of its simplexes. Condition $K1$ ensures that a geometric complex is a union of closed rectilinear simplexes. Condition $K2$ ensures that they fit together in a satisfactory way.

3 Polyhedra

The underlying space of a complex K, that is, the set of points of R^m belonging to some simplex of K, with the topology induced by that in R^m , is called the *polyhedron* of K and written $|K|$. The complex K is said to be a *dissection* or *triangulization* of the polyhedron $|K|$. $|K|$. is more than just a topological space; it inherits from R^m a *metric* and an affine structure in each simplex. Clearly, different complexes may have the same polyhedron. If x is a point of $|K|$, then, by $K2$, it belongs to just one simplex of K which is called the *carrier* of x.

If K is a complex, a subcollection K_o of its simplexes is called a *subcomplex* of K if it is a complex. If K_o is a subcomplex of K, $|K_o|$ is called a *subpolyhedron* of $|K|$. In an obvious sense we may say that K_o is a subcomplex if it is closed, that is, if it is a union of closed simplexes of K. An important example of a subcomplex is the set of all faces of some simplex $s^p$of K; then $|K_o| = \mathring{s}^p$. We may also take all proper faces of s^p; then $|K_o| = s^p$.

Let K^n be the set of simplexes of K of dimension $\leq n$; then K^n is a subcomplex of K, called the *n-section* of K. A complex is a collection of simplexes so that unions and intersections of complexes are again collections of simplexes. The intersection of two complexes in R^m is certainly a complex, but its polyhedron may not be the intersection of the underlying polyhedra. However, if K and L are complexes in R^m and if $|K| \cap |L| = |M|$, where M is the subcomplex of K and of L, then $K \cup L$ is a complex.

4 Regular subdivision

A *vertex set* of a complex K is the set of points of R^m that are vertices of simplexes of K. The set of subsets of the vertex set which span simplexes of K is the *vertex scheme* of the complex K. We emphasize that K is known when we know its vertex set and its vertex scheme. This point of view is useful since a complex is often presented as a vertex scheme.

If K is a geometric complex in R^m , then $|K|$ admits the rectilinear dissection K. We now describe a process whereby, for complexes K, further rectilinear dissections K', $K\text{—}|$, ... of $|K|$ may be found which chop up $|K|$ up more and more finely. Let V be the vertex scheme of K. We shall define a vertex scheme V' which is the vertex scheme of a complex K' such that $|K| = |K'|$. The complex K' is called the *first derived* of K and is said to be obtained from K by *regular subdivision.* Once the vertex scheme V' is defined, readers will have no difficulty in accepting the truth of this statement.

We now define V'. To each simplex $(\mathbf{a}^{i_0}, ..., \mathbf{a}^{i_p})$ of K we define a vertex $\mathbf{b}^{i_0,...,i_p}$ of V' by

$$\mathbf{b}^{i_0,...,i_p} = \frac{1}{p+1 \sum_{k=0}^{p} \mathbf{a}^{i_k}}.$$

Thus $\mathbf{b}^{i_0,...,i_p}$ is the barycentre, or center of gravity, of equal masses placed at vertices $\mathbf{a}^{i_0}, ..., \mathbf{a}^{i_p}$. Clearly $\mathbf{b}^{i_0,...,i_p}$ is symmetric in its superscripts. We next describe the selected subsets of the vertices which are to belong to V'; namely, a subset of the $\mathbf{b}$'s is selected if and only if its elements may be so ordered that the set of superscripts for each vertex is contained in the set of superscripts for the preceding vertex. For instance, if $\mathbf{b}^{1,9,3,4,7}$, $\mathbf{b}^{4,3}$, $\mathbf{b}^{3,9,4}$ are vertices of V', they form a selected subset since they may be ordered as $\mathbf{b}^{1,9,3,4,7}, \mathbf{b}^{3,9,4}, \mathbf{b}^{4,3}$. On the other hand, $\mathbf{b}^{1,3,4,7}, \mathbf{b}^{4,3}, \mathbf{b}^{3,9,4}$ is not selected.

5 The cone construction

Let V be a vertex scheme in R^m and $\mathbf{a}$ a point of R^m which is not a vertex of V. We define $\mathbf{a}V$ to be the vertex scheme whose vertices are those of V with $\mathbf{a}$

adjoined and whose subsets are (i) the selected subsets of V, (ii) $\mathbf{a}$ itself and (iii) all subsets consisting of the union of $\mathbf{a}$ with a selected subset of V. We call $\mathbf{a}V$ the *cone* on V with vertex $\mathbf{a}$. If V is the vertex scheme of a complex $\mathcal{L}$ and $\mathbf{a}V$ is the vertex scheme of a complex $\mathcal{K}$, we call $\mathcal{K}$ the *m-cone* on $\mathcal{L}$ with vertex $\mathbf{a}$ and write $\mathcal{K} = \mathbf{a}\mathcal{L}$, where m is the dimension of $\mathcal{K}$. Thus, for example, if $|\mathcal{L}| \subseteq R^{m-1}$ and R^{m-1} is embedded in R^m , and if $\mathbf{a} \in R^m - R^{m-1}$, the $\mathbf{a}\mathcal{L}$ is a complex and $|\mathbf{a}\mathcal{L}|$ is the cone on the base $|\mathcal{L}|$ with vertex $\mathbf{a}$. Therefore, we can build a cone on the base of a $(p-1)$-simplex to form a p-simplex. Let $\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p$ be an independent set in R^m and let $s^p = (\mathbf{a}^0, \mathbf{a}^1, ..., \mathbf{a}^p)$, $s^{p-1} = (\mathbf{a}^1, ..., \mathbf{a}^p)$, then $\mathring{s}^p = \mathbf{a}^o \mathring{s}^{p-1}$.